NS9360 Hardware Reference

90000675_J

Contents

Using This Guide

Review this section for basic information about the guide you are using, as well as general support and contact information.

About this guide

This guide provides information about the Digi NS9360, a single chip 0.13µm CMOS network-attached processor. The NS9360 is part of the 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 NS9360 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 NS9360 architecture.

What's in this guide

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

To read about	See	Vol
NS9360 key features	Chapter 1, "About the NS9360"	1
NS9360 ball grid array assignments	Chapter 2, "NS9360 Pinout"	1
NS9360 CPU	Chapter 3, "Working with the CPU"	1
System functionality	Chapter 4, "System Control Module"	1
How the NS9360 works with the Multiport Memory Controller, an AMBA-compliant SoC peripheral	Chapter 5, "Memory Controller"	1
How the NS9360 works with Ethernet MAC and Ethernet front-end module	Chapter 6, "Ethernet Communication Module"	1
Digi proprietary BBus	Chapter 7, "BBus Bridge	1
NS9360 BBus DMA controller subsystem	Chapter 8, "BBus DMA Controller"	1
Chip-level support for low-speed peripherals	Chapter 9, "BBus Utility"	1
Real time clock module	Chapter 10, "Real Time Clock Module"	2
Interface between the ARM CPU and the I^2C bus	Chapter 11, "I^2C 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 Host Module" Chapter 17, "USB Device Module"	2
NS9360 electrical characteristics and timing diagrams and information	Chapter 18, "Timing"	2
NS9360 packaging information	Chapter 19, "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

- *NS9360 Jumpers and Components* provides a hardware description of the NS9360 development board, and includes information about jumpers, components, switches, and configuration.
- *NS9360 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 (www.digi.com/support).

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 NS9360

C H A P T E R 1

The NS9360 is a single chip 0.13μm CMOS network-attached processor. This chapter provides an overview of the NS9360, which is based on the standard architecture in the NET+ARM family of devices.

About the NS9360

The NS9360 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 NS9360 runs up to 180 MHz, with a 90 MHz system and memory bus and 45 MHz peripheral bus. The NS9360 offers an extensive set of I/O interfaces and Ethernet high-speed performance and processing capacity. The NS9360 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.

NS9360 Features

32-bit ARM926EJ-S RISC processor

- 103 to 177 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, WinCE/Pocket PC, 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

Flexible LCD controller

- Supports most commercially available displays:
  - 18-bit active Matrix color TFT displays
  - Single and dual panel color STN displays
  - Single and dual panel monochrome STN displays
- 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)
- Independent OHCI Host and Device ports
- Internal USB PHY
- External USB PHY interface
- USB device supports one bidirectional control endpoint and 10 unidirectional endpoints
- All endpoints supported by a dedicated DMA channel
- 32 byte FIFO per endpoint

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 11.25 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^2C port

- I^2C 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^2C 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 12-channel DMA engine supports USB device
  - 2 DMA channels support control endpoint
  - 10 DMA channels support 10 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

- 8 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

- 73 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

- 177 MHz: 0 – 70^{o} C
- 155 MHz: -40 – +85^{o} C
- 103 MHz: 0 – 70^{o} C

System-level interfaces

Figure 1 shows the NS9360 system-level interfaces.

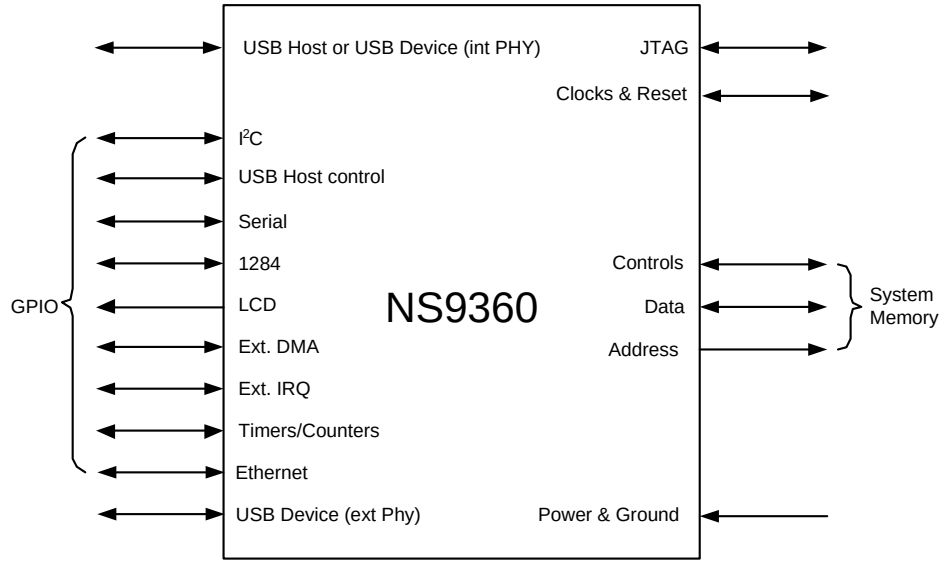


Figure 1: System-level hardware interfaces

- 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
- USB Host or Device interface using internal USB PHY
- I^2C interface
- 73 GPIO pins muxed with:
  - Four 8-pin-each serial ports, each programmable to UART or SPI
  - 1284 port
  - LCD controller interface
  - 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
- Ethernet interface
  - USB Device interface to external USB PHY
- 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 NS9360 system:

- 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 NS9360 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.

Reset

Master reset using an external reset pin resets NS9360. 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 behavior

	RESET_n pin	SRESET_n pin	PLL Config Reg. Update	Watchdog Time-Out Reset
SPI Boot	Yes	Yes	Yes	Yes
Strapping PLL	Yes	No	No	No
Other Strappings (Endianess)	Yes	No	No	No
GPIO Configuration	Yes	No	No	No
Other (ASIC) Registers	Yes	Yes	Yes	Yes

Table 1: Reset behavior

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.

SPI 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.

RESET_DONE as an output

Sets to 1, per Step 6 in the SPI 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.

You can use one of these software resets to reset NS9360. Select the reset by setting the appropriate bit in the appropriate register.

- Watchdog timer can issue reset upon watchdog timer expiration (see the "Software Watchdog Timer register" on page 174).
- Software reset can reset individual internal modules or all modules except memory and CPU (see "Reset and Sleep Control register" on page 177).
- The system is reset whenever software sets the PLL SW change bit, in the PLL Configuration register, to 1.

Hardware reset duration is 4 ms for PLL to stabilize. Software duration depends on speed grade, as shown in Table 2: "Software reset duration" on page 35.

Speed grade	CPU clock cycles	Duration
177 MHz	128	723 ns
155 MHz	128	826 ns
103 MHz	128	1243 ns

Table 2: Software reset duration

The minimum reset pulse width is 10 crystal clocks.

System clock

The system clock is provided to NS9360 by either a crystal or an external oscillator. Table 3 shows sample clock frequency settings for each chip speed grade.

Speed	cpu_clk	ahb_hclk (main bus)	bbus_clk
177 MHz	176.9472	88.4736	44.2368
155 MHz	154.8288	77.4144	38.7072
103 MHz	103.2192	51.6096	24.8048

Table 3: Sample clock frequency settings with 29.4912 MHz crystal

Pulldowns are required as follows:

- To produce 176.9472 MHz, pull down gpio[12], gpio[10], gpio[4].
- To produce 154.8288 MHz, pull down gpio[12], gpio[10], gpio[8].
- To produce 103.2192 MHz, pull down gpio[17], gpio[10], gpio[8], gpio[4].

Using an oscillator

If an oscillator is used, it must be connected to the x1_sys_osc input (C8 pin) on the NS9360. If a crystal is used, it must be connected with a circuit such as the one shown in Figure 2, "System clock".

The PLL parameters are initialized on powerup reset, and can be changed by software. For a 177 MHz grade, the CPU may change from 177 MHz to 103 MHz, the AHB system bus may change from 88 MHz to 51 MHz, and the peripheral BBus may change from 44 MHz to 26 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, 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 V4 for receive clock and tx_clk on pin V2 for transmit clock. For a RMII, there is only one clock, and it connects to the rx_clk on pin V4. In this case, the transmit clock, tx_clk, should be tied low.

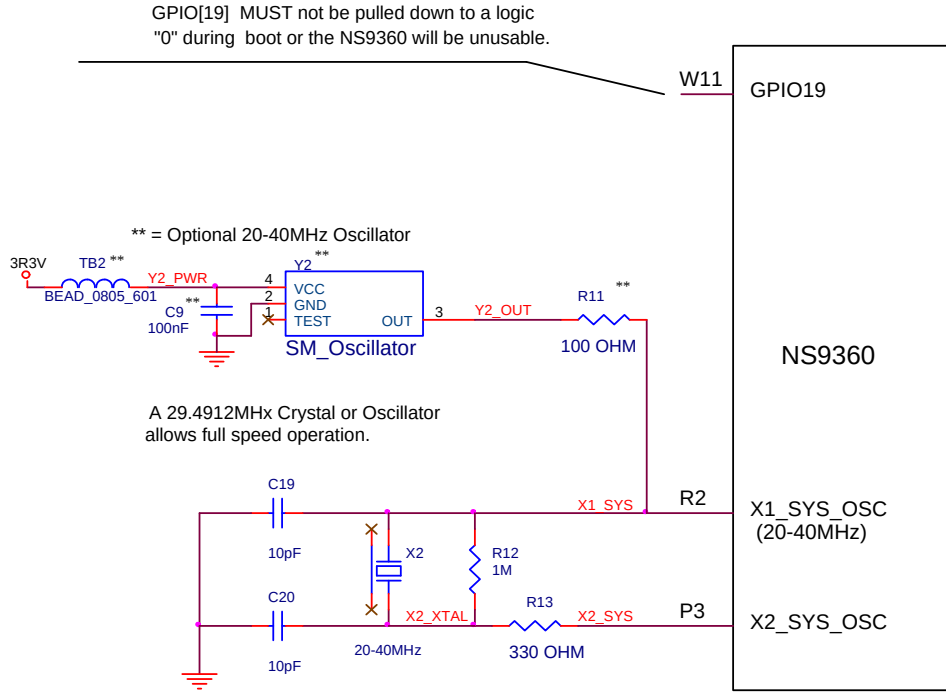


Figure 2: System clock

LCD controller, serial modules (UART, SPI), and the 1284 port optionally can 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.

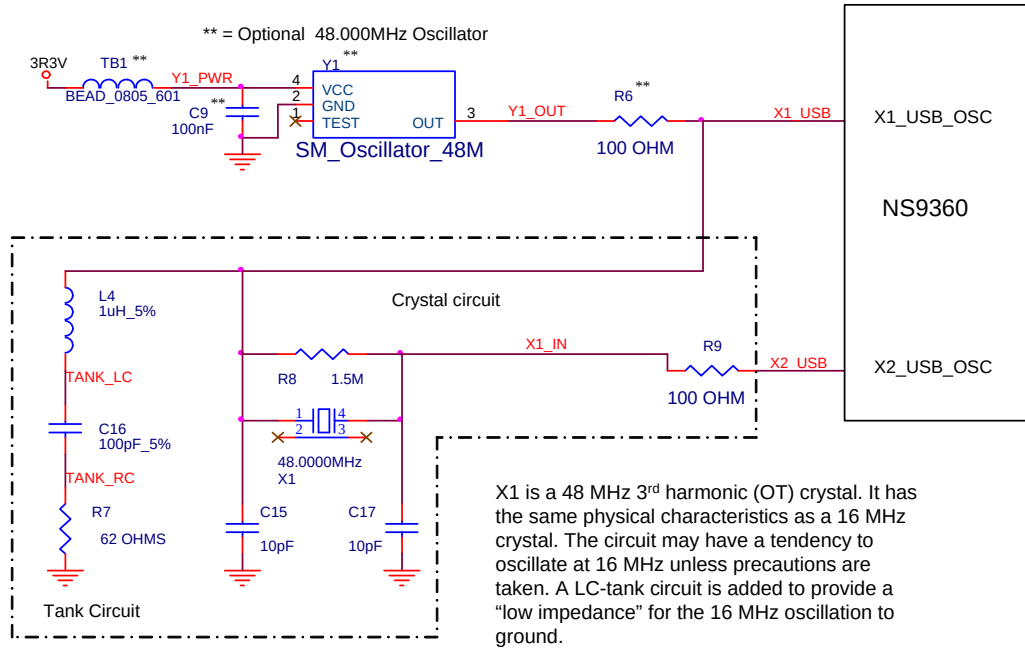


Figure 3: USB clock

NS9360 Pinout

C H A P T E R 2

The NS9360 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 USB port, and four multi-function serial ports. The NS9360 provides up to 73 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 has an internal pullup resistor or a pulldown resistor: ■ **U** — Pullup (input current source) ■ **D** — Pulldown (input current sink) If no value appears, that pin has neither an internal pullup nor pulldown resistor.
I/O	The type of signal: input, output, or input/output.
OD (mA)	The output drive of an output buffer. NS9360 uses one of three drivers: ■ 2 mA ■ 4 mA ■ 8 mA

More detailed signal descriptions are provided for selected modules.

System Memory interface

Note: Some System Memory interface signals are muxed behind gpio. These are noted in the Signal name / muxed behind column. If there is no slash and no gpio pin indicated, the signal is not muxed behind a gpio signal.

Pin #	Signal name / muxed behind	U/D	OD (mA)	I/O	Description
P18	addr[0]		8	O	Address bus signal
R20	addr[1]		8	O	Address bus signal
P19	addr[2]		8	O	Address bus signal
P20	addr[3]		8	O	Address bus signal
N18	addr[4]		8	O	Address bus signal
N19	addr[5]		8	O	Address bus signal
N20	addr[6]		8	O	Address bus signal
M18	addr[7]		8	O	Address bus signal
M19	addr[8]		8	O	Address bus signal
M20	addr[9]		8	O	Address bus signal
L19	addr[10]		8	O	Address bus signal
L18	addr[11]		8	O	Address bus signal
L20	addr[12]		8	O	Address bus signal
K20	addr[13]		8	O	Address bus signal
K18	addr[14]		8	O	Address bus signal
K19	addr[15]		8	O	Address bus signal
J20	addr[16]		8	O	Address bus signal
J19	addr[17]		8	O	Address bus signal
J18	addr[18]		8	O	Address bus signal
H20	addr[19]		8	O	Address bus signal
H18	addr[20]		8	O	Address bus signal
G20	addr[21]		8	O	Address bus signal

Table 4: System Memory interface pinout

Pin #	Signal name / muxed behind	U/D	OD (mA)	I/O	Description
H19	addr[22] / gpio[66]		8	O	Address bus signal
E18	addr[23] / gpio[67]		8	O	Address bus signal
D19	addr[24] / gpio[68]		8	O	Address bus signal
C20	addr[25] / gpio[69]		8	O	Address bus signal
A17	addr[26] / gpio[70]		8	O	Address bus signal
B16	addr[27] / gpio[71]		8	O	Address bus signal
D19	clk_en[0] / gpio[68]		8	O	SDRAM clock enable
C20	clk_en[1] / gpio[69]		8	O	SDRAM clock enable
A17	clk_en[2] / gpio[70]		8	O	SDRAM clock enable
B16	clk_en[3] / gpio[71]		8	O	SDRAM clock enable
C15	clk_out[0]		8	O	SDRAM clock
A12	clk_out[1]		8	O	SDRAM reference clock. Connect to clk_in using AC termination.
A7	clk_out[2]		8	O	SDRAM clock
G1	clk_out[3]		8	O	SDRAM clock
A16	data[0]		8	I/O	Data bus signal
B15	data[1]		8	I/O	Data bus signal
C14	data[2]		8	I/O	Data bus signal
A15	data[3]		8	I/O	Data bus signal
B14	data[4]		8	I/O	Data bus signal
A14	data[5]		8	I/O	Data bus signal
C13	data[6]		8	I/O	Data bus signal
B13	data[7]		8	I/O	Data bus signal
A13	data[8]		8	I/O	Data bus signal
C12	data[9]		8	I/O	Data bus signal
B12	data[10]		8	I/O	Data bus signal
B11	data[11]		8	I/O	Data bus signal

Table 4: System Memory interface pinout

Pin #	Signal name / muxed behind	U/D	OD (mA)	I/O	Description
C11	data[12]		8	I/O	Data bus signal
A11	data[13]		8	I/O	Data bus signal
A10	data[14]		8	I/O	Data bus signal
C10	data[15]		8	I/O	Data bus signal
B10	data[16]		8	I/O	Data bus signal
A9	data[17]		8	I/O	Data bus signal
B9	data[18]		8	I/O	Data bus signal
C9	data[19]		8	I/O	Data bus signal
A8	data[20]		8	I/O	Data bus signal
B8	data[21]		8	I/O	Data bus signal
C8	data[22]		8	I/O	Data bus signal
B7	data[23]		8	I/O	Data bus signal
A6	data[24]		8	I/O	Data bus signal
C7	data[25]		8	I/O	Data bus signal
B6	data[26]		8	I/O	Data bus signal
A5	data[27]		8	I/O	Data bus signal
C6	data[28]		8	I/O	Data bus signal
B5	data[29]		8	I/O	Data bus signal
A4	data[30]		8	I/O	Data bus signal
C5	data[31]		8	I/O	Data bus signal
F2	data_mask[0]		8	O	SDRAM data mask signal
G3	data_mask[1]		8	O	SDRAM data mask signal
F1	data_mask[2]		8	O	SDRAM data mask signal
G2	data_mask[3]		8	O	SDRAM data mask signal
J2	clk_in			I	SDRAM feedback clock. Connect to clk_out[1].
D1	byte_lane_sel_n[0]		8	O	Static memory byte_lane_enable[0] or write_enable_n[0] for byte-wide device signals

Table 4: System Memory interface pinout

Pin #	Signal name / muxed behind	U/D	OD (mA)	I/O	Description
E2	byte_lane_sel_n[1]		8	O	Static memory byte_lane_enable[1] or write_enable_n[1] for byte-wide device signals
F3	byte_lane_sel_n[2]		8	O	Static memory byte_lane_enable[2] or write_enable_n[2] for byte-wide device signals
E1	byte_lane_sel_n[3]		8	O	Static memory byte_lane_enable[3] or write_enable_n[3] for byte-wide device signals
H1	cas_n		8	O	SDRAM column address strobe
B4	dy_cs_n[0]		8	O	SDRAM chip select signal
A3	dy_cs_n[1]		8	O	SDRAM chip select signal
D5	dy_cs_n[2]		8	O	SDRAM chip select signal
C4	dy_cs_n[3]		8	O	SDRAM chip select signal
H2	st_oe_n		8	O	Static memory output enable
J3	ras_n		8	O	SDRAM row address strobe
B3	st_cs_n[0]		8	O	Static memory chip select signal
C1	st_cs_n[1]		8	O	Static memory chip select signal
D2	st_cs_n[2]		8	O	Static memory chip select signal
E3	st_cs_n[3]		8	O	Static memory chip select signal
H3	we_n		8	O	SDRAM write enable. Used for static and SDRAM devices.
J1	ta_strb / gpio[72]			I	Slow peripheral transfer acknowledge

Table 4: System Memory interface pinout

System Memory interface signals

Table 5 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 [27:0]. **Note:** Address bits [27:22] are muxed behind gpio[71:66].
clk_en[3:0]	O	SDRAM clock enable. Used for SDRAM devices. These signals are muxed behind gpio[71:68]. **Note:** The clk_en signals are associated with the dy_cs_n signals. If clock enables are used, a pullup resistor is required to prevent floating during startup, and to avoid SDRAM lockup during manual or brown out conditions. If not used, connect the clock enables in the SDRAM devices directly to 3.3v or a pullup resistor.
clk_out[3:0]	O	SDRAM clocks. Used for SDRAM devices. SDRAM clk_out[1] is connected to clk_in.
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	I	Feedback clock. Always connects to clk_out[1].
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. This signal is muxed behind gpio[72].

Table 5: System Memory interface signal descriptions

Figure 4 shows an example of NS9360 SDRAM clock termination. clk_out[1] is shown, but you can use any clk_out signal (0, 1, 2, or 3).

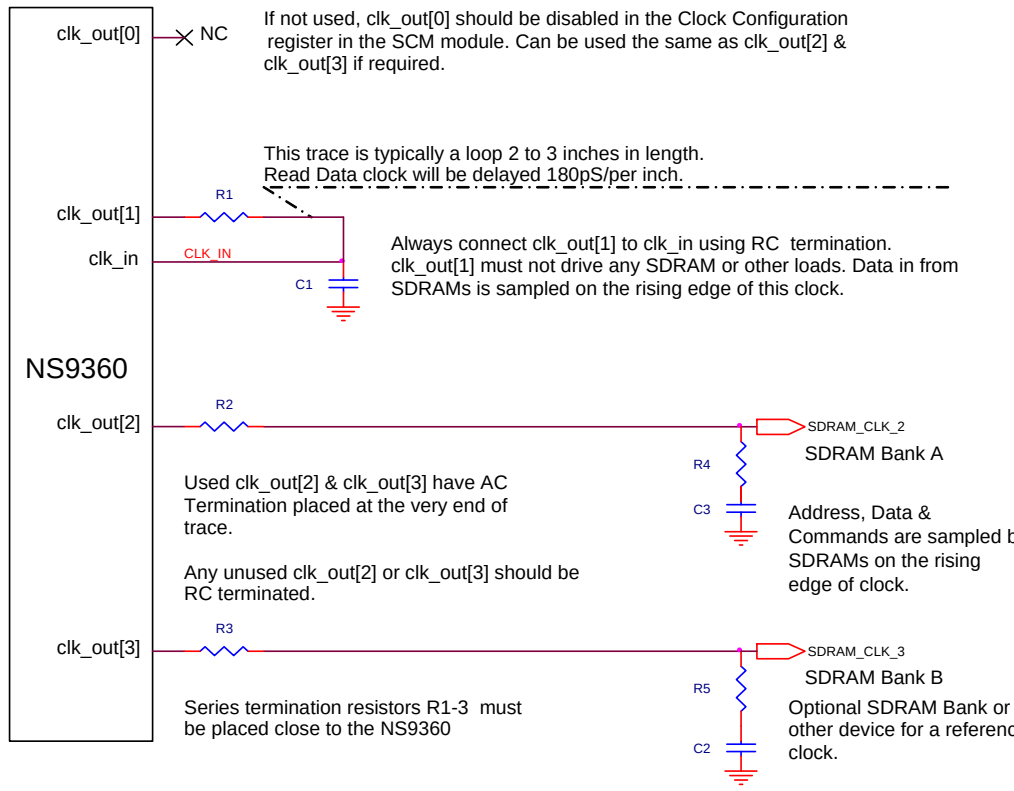


Figure 4: SDRAM clock termination

Ethernet interface

Notes:

- Most Ethernet MII signals are muxed behind gpio. These are noted in the Signal name: MII / muxed behind column. If there is no slash and no gpio pin indicated, the Ethernet signal is not muxed behind a gpio signal.
- N/C indicates *No Connect* or *ground*, as indicated in the description for the pin.

Pin #	Signal name		U/D	OD (mA)	I/O	Description	
	MII / muxed behind	RMII				MII	RMII
P2	col / gpio[63]	N/C			I	Collision	Pull low external to NS9360
R1	crs / gpio[64]	crs_dv			I	Carrier sense	Carrier sense/data valid
P1	enet_phy_int_n / gpio[65]	enet_phy_int_n	U		I	Ethernet PHY interrupt	Ethernet PHY interrupt
L2	mdc	mdc		4	O	MII management interface clock	MII management interface clock
K2	mdio / gpio[50]	mdio		2	I/O	MII management data	MII management data
V4	rx_clk	ref_clk			I	Receive clock	Reference clock
U3	rx_dv / gpio[51]	N/C			I	Receive data valid	Pull low external to NS9360
V1	rx_er / gpio[52]	rx_er			I	Receive error	Optional signal; pull low external to NS9360 if not used
N3	rxd[0] / gpio[53]	rxd[0]			I	Receive data bit 0	Receive data bit 0
N2	rxd[1] / gpio[54]	rxd[1]			I	Receive data bit 1	Receive data bit 1
N1	rxd[2] / gpio[55]	N/C			I	Receive data bit 2	Pull low external to NS9360
M3	rxd[3] / gpio[56]	N/C			I	Receive data bit 3	Pull low external to NS9360
V2	tx_clk	N/C			I	Transmit clock	Pull low external to NS9360
M2	tx_en / gpio[57]	tx_en		2	O	Transmit enable	Transmit enable
M1	tx_er / gpio[58]	N/C		2	O	Transmit error	N/A

Table 6: Ethernet interface pinout

Pin #	Signal name		U/D	OD (mA)	I/O	Description	
	MII / muxed behind	RMII				MII	RMII
L3	txd[0] / gpio[59]	txd[0]		2	O	Transmit data bit 0	Transmit data bit 0
L1	txd[1] / gpio[60]	txd[1]		2	O	Transmit data bit 1	Transmit data bit 1
K1	txd[2] / gpio[61]	N/C		2	O	Transmit data bit 2	N/A
K3	txd[3] / gpio[62]	N/C		2	O	Transmit data bit 3	N/A

Table 6: Ethernet interface pinout

Clock generation/system pins

Pin #	Signal name	U/D	OD (mA)	I/O	Description
R2	x1_sys_osc			I	System clock crystal oscillator circuit input
P3	x2_sys_osc			O	System clock crystal oscillator circuit output
T1	sys_osc_vdd				System oscillator 3.3V power
F18	x1_usb_osc			I	USB clock crystal oscillator circuit input. (Connect to GND if USB is not used.)
E20	x2_usb_osc			O	USB clock crystal oscillator circuit output
E19	usb_osc_vdd				USB oscillator 3.3V power
W4	reset_done	U	2	I/O	CPU is enabled once the boot program is loaded. Reset_done is set to 1.
U5	reset_n	U		I	System reset input signal
W3	sreset_n	U		I	System reset. sreset_n is the same as reset but does *not* reset the system PLL.
V5	bist_en_n			I	Enable internal BIST operation

Table 7: Clock generation and system pin pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
U6	pll_test_n			I	Enable PLL testing
Y3	scan_en_n			I	Enable internal scan testing
R3	sys_pll_dvdd				System clock PLL 1.5V digital power
T2	sys_pll_dvss				System clock PLL digital ground
U2	sys_pll_avdd				System clock PLL 3.3V analog power
U1	sys_pll_avss				System clock PLL analog ground
T3	pll_lpf			O	PLL diagnostic output
V10	lcdclk / gpio[15]	U		I	External LCD clock input (muxed behind gpio[15])

Table 7: Clock generation and system pin pinout

bist_en_n, pll_test_n, and scan_en_n

Table 8 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 8: bist_en_n, pll_test_n, & scan_en_n truth/termination table

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 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) are 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*.
- The I^2C module must be held in reset until the GPIO assigned to I^2C has been configured.
- GPIO pins 42, 43, 50-64, and 66-72 (24 pins total) do not have internal pullups. All GPIO pins are reset at powerup to be inputs. Any of these pins not being used should be pulled high with an external 10K resistor. Some of these GPIO pins might need external 10K pullup or pulldown resistors to prevent them from floating before being programmed, depending on how they are being used.

  For example, gpio[66] is being used as *mem addr [22]* to address a flash chip. gpio[66] should be pulled down with a 10K resistor to prevent the pin from floating until the software is loaded and can program the pin using GPIO Configuration Register 9 to drive *mem addr [22]* from pin gpio[66].

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
W5	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

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
V6	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
Y5	gpio [2][1]	U	2	I/O	00	Ser port B RTS
					01	Timer 0
					02	DMA ch 2 read enable
					03	GPIO 2
W6	gpio[3]	U	2	I/O	00	Ser port B CTS
					01	1284 nACK (peripheral-driven)
					02	DMA ch 1 req
					03	GPIO 3
V7	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
Y6	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
W7	gpio[6]	U	2	I/O	00	Ser port B RI / SPI port B clk
					01	1284 nFault (peripheral-driven)
					02	Timer 7 (duplicate)
					03	GPIO 6
Y7	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
V8	gpio[8][1]	U	2	I/O	00	Ser port A TxData / SPI port A dout
					01	Reserved
					02	Reserved
					03	GPIO 8

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
W8	gpio[9]	U	2	I/O	00	Ser port A RxData / SPI port A din
					01	Reserved
					02	Reserved
					03	GPIO 9
Y8	gpio[10][1]	U	2	I/O	00	Ser port A RTS
					01	Reserved
					02	PWM ch 0 (duplicate)
					03	GPIO 10
V9	gpio[11]	U	2	I/O	00	Ser port A CTS
					01	Ext IRQ2 (duplicate)
					02	Timer 0 (duplicate)
					03	GPIO 11
W9	gpio[12][1]	U	2	I/O	00	Ser port A DTR
					01	Reserved
					02	PWM ch 1 (duplicate)
					03	GPIO 12
Y9	gpio[13]	U	2	I/O	00	Ser port A DSR
					01	Ext IRQ 0 (duplicate)
					02	PWM ch 2 (duplicate)
					03	GPIO 13
W10	gpio[14]	U	2	I/O	00	Ser port A RI / SPI port A clk
					01	Timer 1
					02	PWM ch 3 (duplicate)
					03	GPIO 14
V10	gpio[15]	U	2	I/O	00	Ser port A DCD / SPI port A enable
					01	Timer 2
					02	LCD clock input
					03	GPIO 15
Y10	gpio[16][2]	U	2	I/O	00	USB overcurrent
					01	1284 nFault (peripheral-driven, duplicate)
					02	Reserved
					03	GPIO 16

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
Y11	gpio[17][1,2]	U	2	I/O	00	USB power relay
					01	Reserved
					02	Reserved
					03	GPIO 17
V11	gpio[18]	U	4	I/O	00	Ethernet CAM reject
					01	LCD power enable
					02	Ext IRQ 3 (duplicate)
					03	GPIO 18
W11	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
Y12	gpio[20][1]	U	8	I/O	00	Ser port C DTR
					01	LCD clock
					02	Reserved
					03	GPIO 20
W12	gpio[21]	U	4	I/O	00	Ser port C DSR
					01	LCD frame pulse-vert
					02	Reserved
					03	GPIO 21
V12	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
Y13	gpio[23]	U	4	I/O	00	Ser port C DCD / SPI port C enable
					01	LCD line end
					02	Reserved
					03	GPIO 23
W13	gpio[24][1]	U	4	I/O	00	Ser port D DTR
					01	LCD data bit 0
					02	Reserved
					03	GPIO 24

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
V13	gpio[25]	U	4	I/O	00	Ser port D DSR
					01	LCD data bit 1
					02	Reserved
					03	GPIO 25
Y14	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
W14	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
Y15	gpio[28]	U	4	I/O	00	Ext IRQ 1 (duplicate)
					01	LCD data bit 4
					02	LCD data bit 8 (duplicate)
					03	GPIO 28
V14	gpio[29]	U	4	I/O	00	Timer 5
					01	LCD data bit 5
					02	LCD data bit 9 (duplicate)
					03	GPIO 29
W15	gpio[30]	U	4	I/O	00	Timer 6
					01	LCD data bit 6
					02	LCD data bit 10 (duplicate)
					03	GPIO 30
Y16	gpio[31]	U	4	I/O	00	Timer 7
					01	LCD data bit 7
					02	LCD data bit 11 (duplicate)
					03	GPIO 31
V15	gpio[32]	U	4	I/O	00	Ext IRQ 2
					01	1284 Data 1 (bidirectional)
					02	LCD data bit 8
					03	GPIO 32

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
W16	gpio[33]	U	4	I/O	00	Reserved
					01	1284 Data 2 (bidirectional)
					02	LCD data bit 9
					03	GPIO 33
Y17	gpio[34]	U	4	I/O	00	iic_scl
					01	1284 Data 3 (bidirectional)
					02	LCD data bit 10
					03	GPIO 34
U15	gpio[35]	U	4	I/O	00	iic_sda
					01	1284 Data 4 (bidirectional)
					02	LCD data bit 11
					03	GPIO 35
V16	gpio[36]	U	4	I/O	00	PWM ch 0
					01	1284 Data 5 (bidirectional)
					02	LCD data bit 12
					03	GPIO 36
W17	gpio[37]	U	4	I/O	00	PWM ch 1
					01	1284 Data 6 (bidirectional)
					02	LCD data bit 13
					03	GPIO 37
Y18	gpio[38]	U	4	I/O	00	PWM ch 2
					01	1284 Data 7 (bidirectional)
					02	LCD data bit 14
					03	GPIO 38
U16	gpio[39]	U	4	I/O	00	PWM ch 3
					01	1284 Data 8 (bidirectional)
					02	LCD data bit 15
					03	GPIO 39
V17	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

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
W18	gpio[41]	U	4	I/O	00	Ser port C RxData / SPI port C din
					01	Reserved
					02	LCD data bit 17
					03	GPIO 41
U18	gpio[42]		2	I/O	00	Ser port C RTS
					01	Reserved
					02	USB phy data + (TX and RX data for bidirectional PHY or TX data only for unidirectional PHY)
					03	GPIO 42
V20	gpio[43]		2	I/O	00	Ser port C CTS
					01	1284 transceiver direction control
					02	USB phy data - (TX and RX data for bidirectional PHY or TX data only for unidirectional PHY)
					03	GPIO 43
U19	gpio[44][1]	U	2	I/O	00	Ser port D TxData / SPI port D dout
					01	1284 Select (peripheral-driven)
					02	USB phy tx output enable
					03	GPIO 44
U20	gpio[45]	U	2	I/O	0	Ser port D RxData / SPI port D din
					01	1284 nStrobe (host-driven)
					02	USB phy rx data
					03	GPIO 45
T19	gpio[46]	U	2	I/O	00	Ser port D RTS
					01	1284 nAutoFd (host-driven)
					02	USB phy rx data + (unidirectional phy only; for bidirectional USB PHY applications, do not configure for option 02)
					03	GPIO 46

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
R18	gpio[47]	U	2	I/O	00	Ser port D CTS
					01	1284 nInit (host-driven)
					02	USB phy rx data - (unidirectional phy only; for bidirectional USB PHY applications, do not configure for option 02)
					03	GPIO 47
T20	gpio[48]	U	2	I/O	00	USB phy suspend
					01	1284 nSelectIn (host-driven)
					02	DMA ch 2 req
					03	GPIO 48
R19	gpio[49][1]	U	2	I/O	00	USB phy speed
					01	1284 periphal logic high (peripheral-driven)
					02	DMA ch 2 done
					03	GPIO 49
K2	gpio[50]		2	I/O	00	MII/RMII management data
					01	USB phy data + (duplicate) (TX and RX data for bidirectional PHY or TX data only for unidirectional PHY)
					02	Reserved
					03	GPIO 50
U3	gpio[51]		2	I/O	00	MII rx data valid
					01	USB phy data - (duplicate) (TX and RX data for bidirectional PHY or TX data only for unidirectional PHY)
					02	Reserved
					03	GPIO 51
V1	gpio[52]		2	I/O	00	MII rx error
					01	USB phy tx output enable (duplicate)
					02	Reserved
					03	GPIO 52
N3	gpio[53]		2	I/O	00	MII/RMII rx data bit 0
					01	USB phy rx data (duplicate)
					02	Reserved
					03	GPIO 53

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
N2	gpio[54]		2	I/O	00	MII/RMII rx data bit 1
					01	USB phy suspend (duplicate)
					02	Reserved
					03	GPIO 54
N1	gpio[55]		2	I/O	00	MII rx data bit 2
					01	USB phy speed (duplicate)
					02	Reserved
					03	GPIO 55
M3	gpio[56]		2	I/O	00	MII rx data bit 3
					01	USB rx data + (duplicate) (unidirectional phy only; for bidirectional USB PHY applications, do not configure for option 01)
					02	Reserved
					03	GPIO 56
M2	gpio[57]		2	I/O	00	MII/RMII tx enable
					01	USB rx data - (duplicate) (unidirectional phy only; for bidirectional USB PHY applications, do not configure for option 01)
					02	Reserved
					03	GPIO 57
M1	gpio[58]		2	I/O	00	MII tx error
					01	Reserved
					02	Reserved
					03	GPIO 58
L3	gpio[59]		2	I/O	00	MII/RMII tx data bit 0
					01	Reserved
					02	Reserved
					03	GPIO 59
L1	gpio[60]		2	I/O	00	MII/RMII tx data bit 1
					01	Reserved
					02	Reserved
					03	GPIO[60]

Table 9: GPIO MUX pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
K1	gpio[61]		2	I/O	00	MII tx data bit 2
					01	Reserved
					02	Reserved
					03	GPIO 61
K3	gpio[62]		2	I/O	00	MII tx data bit 3
					01	Reserved
					02	Reserved
					03	GPIO 62
P2	gpio[63]		2	I/O	00	MII collision
					01	Reserved
					02	Reserved
					03	GPIO 63
R1	gpio[64]		2	I/O	00	MII/RMII carrier sense
					01	Reserved
					02	Reserved
					03	GPIO 64
P1	gpio[65]	U	2	I/O	00	MII/RMII enet phy interrupt
					01	Reserved
					02	Reserved
					03	GPIO 65
H19	gpio[66]		8	I/O	00	Mem addr[22]
					01	Reserved
					02	Reserved
					03	GPIO 66
E18	gpio[67]		8	I/O	00	Mem addr[23]
					01	Reserved
					02	Reserved
					03	GPIO 67
D19	gpio[68]		8	I/O	00	Mem addr[24]
					01	Mem clk_en[0]
					02	Ext IRQ 0 (duplicate)
					03	GPIO 68

Table 9: GPIO MUX pinout

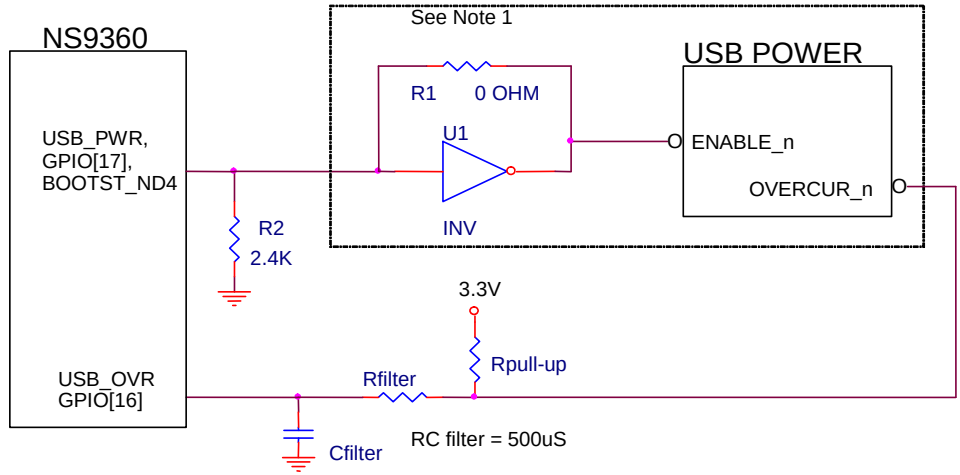
Pin #	Signal name	U/D	OD (mA)	I/O	Description (4 options: 00, 01, 02, 03)	
C20	gpio[69]		8	I/O	00	Mem addr[25]
					01	Mem clk_en[1]
					02	Ext IRQ 1 (duplicate)
					03	GPIO 69
A17	gpio[70]		8	I/O	00	Mem addr[26]
					01	Mem clk_en[2]
					02	iic_scl (duplicate)
					03	GPIO 70
B16	gpio[71]		8	I/O	00	Mem addr[27]
					01	Mem clk_en[3]
					02	iic_sda (duplicate)
					03	GPIO 71
J1	gpio[72]		8	I/O	00	Mem ta_strb
					01	Reserved
					02	Reserved
					03	GPIO 72

1 This pin is used for bootstrap initialization (see Table 49: "Configuration pins — Bootstrap initialization" on page 154). Note that the GPIO pins used as bootstrap pins have a defined powerup state that is required for the appropriate NS9360 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 bootstrap 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.

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 61 for an illustration.

Table 9: GPIO MUX pinout

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



1 Powerup: GPIO[17] = Bootstrap ND4. Can be high or pulled low depending on required CPU speed.
   - If pulled low R2 in; populate inverter U1
   - If not pulled low; populate R1

   R1 and U1 can be eliminated by selecting the ENABLE_n polarity of the USB power IC to match the bootstrap state.

2 Code initializes USB registers. USB_PWR driven by USB IP.

3 Code sets gpio[16] and gpio[17] to mode 0 — USB.
   - Sets the INV function for USB_OVR;
   - If R2 and U1 are populated, set the INV function for USB_PWR

LCD module signals

The LCD module signals are multiplexed with GPIO pins. They include seven control signals and up to 18 data signals. Table 10 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[17:0]	Output	LCD panel data
CLLE	Output	Line end signal

Table 10: LCD module signal descriptions

The CLD[17:0] signal has seven modes of operation:

- 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.

I²C interface

Note: The I²C signals are muxed behind gpio, as noted in the Signal name / muxed behind column.

Pin #	Signal name / muxed behind	U/D	OD (mA)	I/O	Description
Y17	iic_scl/gpio[34]]	U	4	I/O	I²C serial clock line. Add a 4.7K resistor to VDDA (3.3V) for low speed applications, a 1.5K resistor to VDDA (3.3V) for fast mode applications
U15	iic_sda/gpio[35]]	U	4	I/O	I²C serial data line. Add a 4.7K resistor to VDDA (3.3V) for low speed applications, a 1.5K resistor to VDDA (3.3V) for fast mode applications
A17	iic_scl/gpio[70]		8	I/O	I²C serial clock line. Add a 4.7K resistor to VDDA (3.3V) for low speed applications, a 1.5K resistor to VDDA (3.3V) for fast mode applications
B16	iic_sda/gpio[71]		8	I/O	I²C serial data line. Add a 4.7K resistor to VDDA (3.3V) for low speed applications, a 1.5K resistor to VDDA (3.3V) for fast mode applications

Table 11: I²C 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.

Pin #	Signal name	U/D	OD (mA)	I/O	Description
B17	usb_dm			I/O	USB data -

Table 12: USB interface pinout

Pin #	Signal name	U/D	OD (mA)	I/O	Description
A18	usb_dp			I/O	USB data +

Table 12: 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.

Pin #	Signal name	U/D	OD (mA)	I/O	Description
G18	tck			I	Test clock
D20	tdi	U		I	Test data in
G19	tdo		2	O	Test data out
F19	tms	U		I	Test mode select
F20	trst_n	U		I	Test mode reset
Y4	rtck	U	2	I/O	Returned test clock, ARM core only

Table 13: JTAG interface/boundary scan pinout

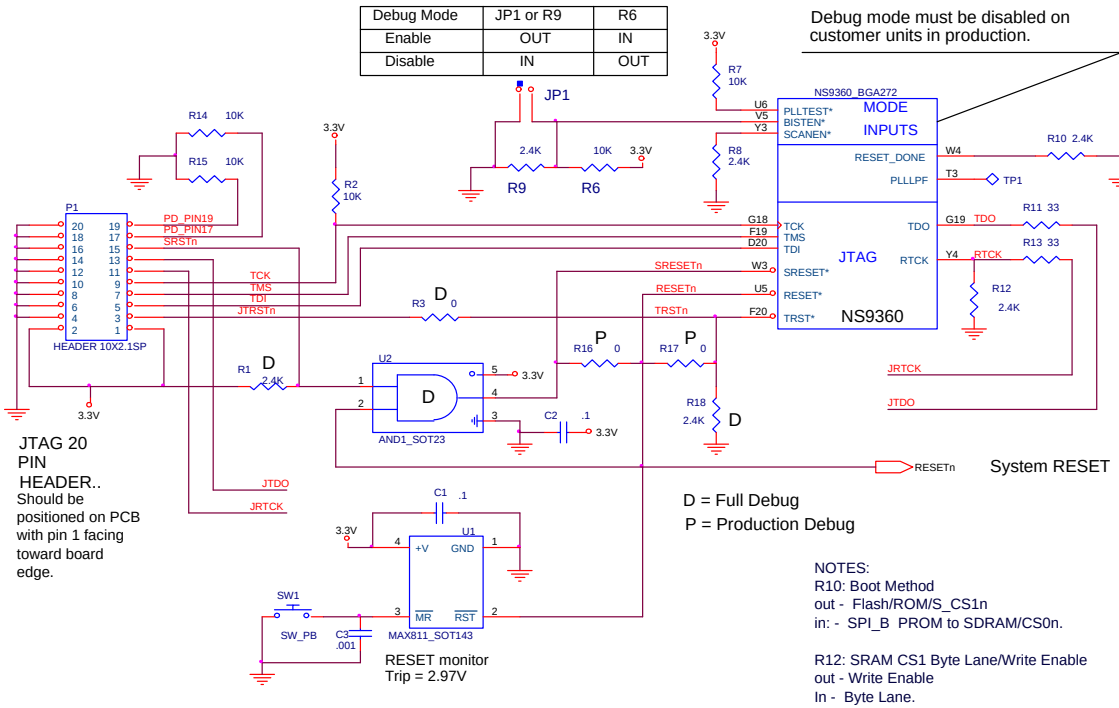


Figure 5: JTAG interface

Reserved

Pin#	Description
T18	No connect

Table 14: Reserved pins

Power and ground

Pin #	Signal name	Description
A2, A1, B2, B1, C3, D4, W1, Y1, W2, Y2, V3, U4, Y19, Y20, W19, W20, V18, U17, B20, A20, B19, A19, C18, D17, E4, H4, J4, C16, H17, E17, M17, N17, T17, D13, D12, D9, D8, U12, U9, T4, N4, C2, D3, D16, C17, D18, B18, C19, V19, J9, J10, J11, J12, K9, K10, K11, K12, L9, L10, L11, L12, M9, M10, M11, M12	VSS	Ground
F4, G4, P4, R4, U8, U7, U13, U14, P17, R17, F17, G17, D15, D14, D7, D6	VDDS	+3.3V
K4, L4, M4, U10, U11, J17, K17, L17, D11, D10	VDDC	+1.5V

Table 15: Power and ground pins

Working with the CPU

C H A P T E R 3

The NS9360 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 98, for information)
- Separate instruction and data AMBA AHB bus interfaces

Figure 6 shows the main blocks in the ARM926EJ-S processor.

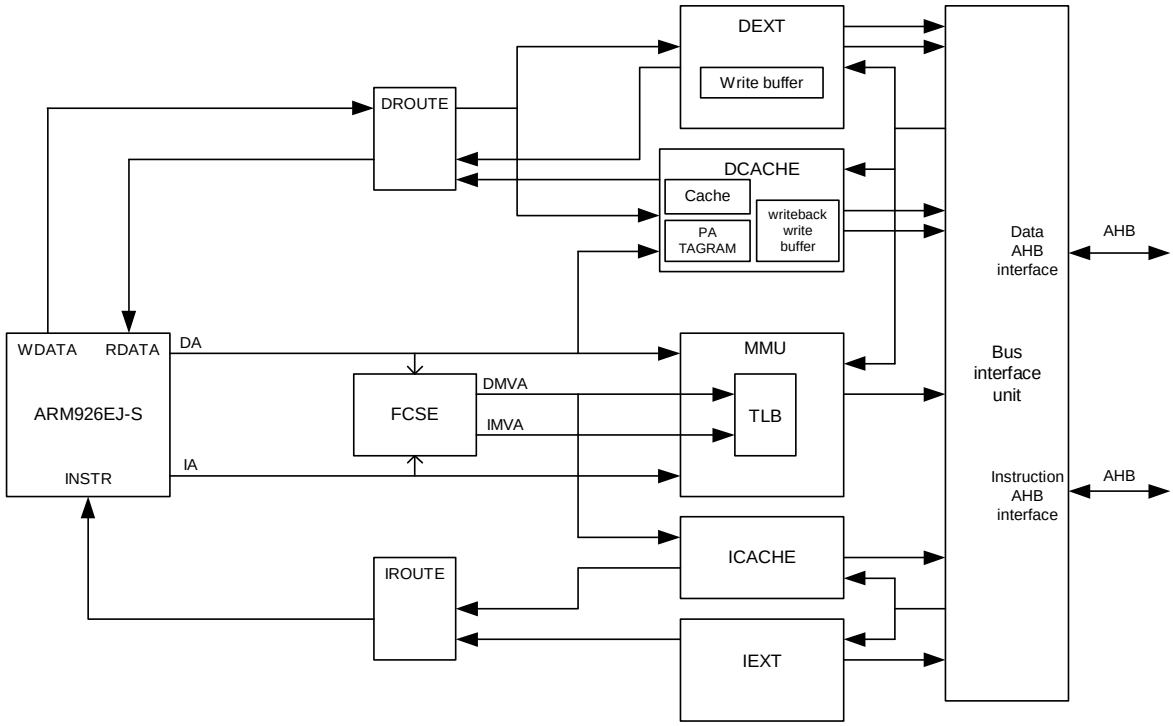


Figure 6: ARM926EJ-S processor block diagram

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 97 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 94).
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 7 shows the MRC and MCR instruction bit pattern.

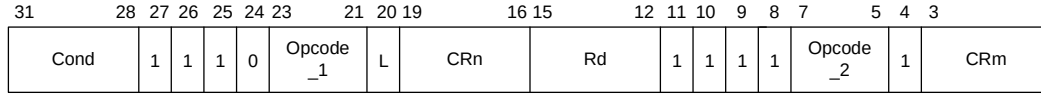


Figure 7: CP15 MRC and MCR bit pattern

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 16 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 16: 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 16: 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 72).
- 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 72).

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 17: 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 17: 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 18 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 18: 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 19 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 19: 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 124 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 8 shows the format of the cache type register. Table 20 describes the fields in the register.

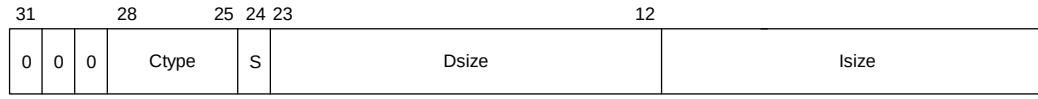


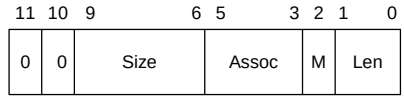
Figure 8: Cache type register format

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 84) Cache lockdown: Format C (see "R9: Cache Lockdown register" on page 89)
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 NS9360.
Dsize	Specifies the size, line length, and associativity of the DCache.
Isize	Species the size, length and associativity of the ICache.

Table 20: 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 NS9360 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 9 shows the Control register format. Table 21 describes the Control register bit functionality.

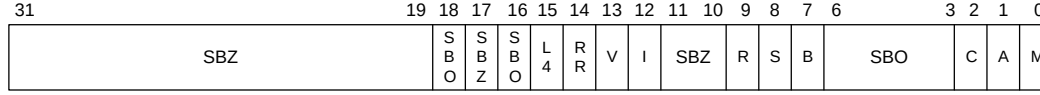


Figure 9: Control register format

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 21: 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 98.
[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 21: R1: Control register bit definition

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

Table 21: 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 22: 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 10 shows the format of the Translation Table Base register.

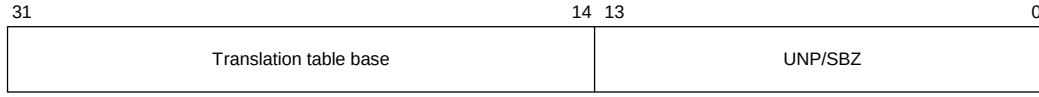


Figure 10: R2: Translation Table Base register

R3: Domain Access Control register

Register R3 is the Domain Access Control register and consists of 16 two-bit fields, as shown in Figure 11.

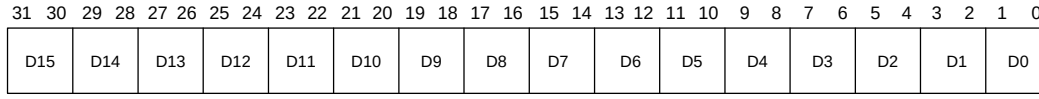


Figure 11: R3: Domain Access Control register

- 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 98, 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 12 shows the format of the Fault Status registers. Table 23 describes the Fault Status register bits.

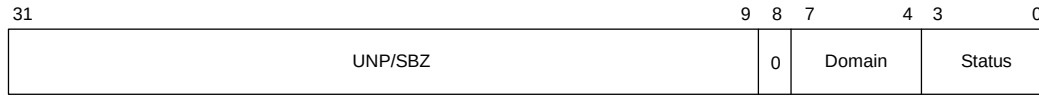


Figure 12: Fault Status registers format

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 98.)

Table 23: Fault Status register bit description

Table 24 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 114 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
Lowest	External abort	Section page	0b1000 0b1010	Valid Valid

Table 24: 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 26, “R7: Cache operations,” on page 86 and "Test and clean operations" on page 87).

Use this instruction to write to the Cache Operations register:

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

Table 25 describes the cache functions provided by register R7. Table 26 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.

Table 25: Cache Operations register function descriptions

Function	Description
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 87).
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 87).
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).
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 25: 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 26: R7: Cache operations

Figure 13 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.

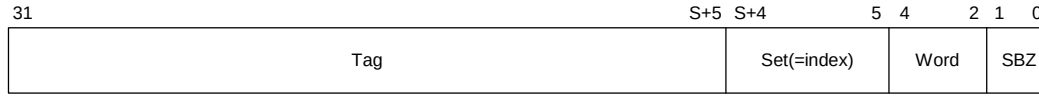


Figure 13: R7: MVA format

Figure 14 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

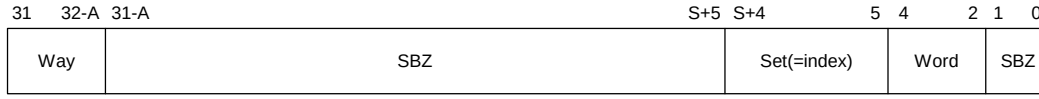


Figure 14: R7: Set/Way format

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 27 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 27: 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 92, for an explanation of how to preserve TLB entries.

Figure 15 shows the modified virtual address format used for invalid TLB single entry operations.

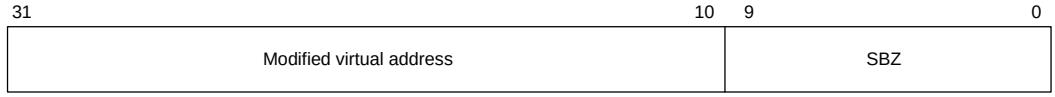


Figure 15: R8: TLB Operations, MVA format

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 29, “Cache Lockdown register L bits,” on page 91). 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 28 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 28: 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 16 shows the format for the Cache Lockdown register.

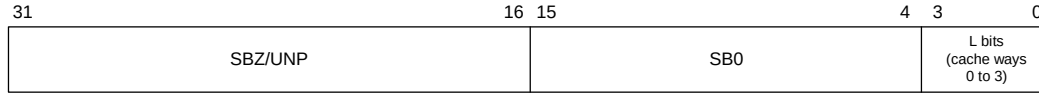


Figure 16: R9: Cache Lockdown register format

Table 29 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 29: 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. Ensure 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 123, for more information).

Figure 17 shows the TLB lockdown format.

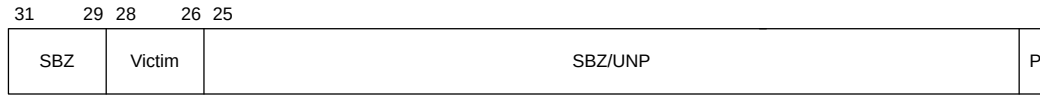


Figure 17: TLB Lockdown register format

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 88 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 18 shows the format of the FCSE PID register.

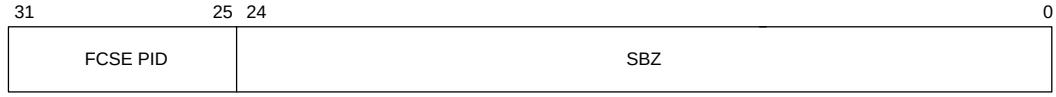


Figure 18: Process ID register format

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

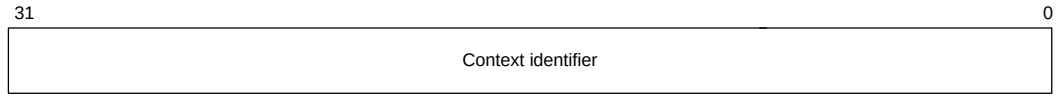


Figure 19: Context ID register format

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 88).
- Invalidate TLB entry selected by MVA, using R8: TLB Operations register (see "R8:TLB Operations register" on page 88).
- Lockdown of TLB entries using R10: TLB Lockdown register (see "R10: TLB Lockdown register" on page 92).

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 81).

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 30 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 31, “Priority encoding of fault status,” on page 104).
R6: Fault Address register	[31:0]	Holds the MVA associated with the access that caused the data abort. See Table 31, “Priority encoding of fault status,” on page 104 for details of the address stored for each type of fault.
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 30: 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 20 shows the format of the TTB register.

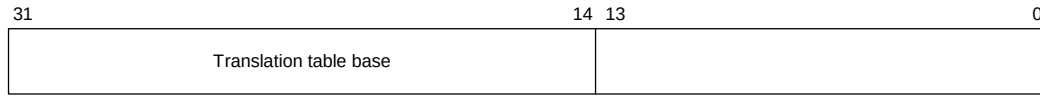


Figure 20: R2: Translation Table base register

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 21 shows the table walk process.

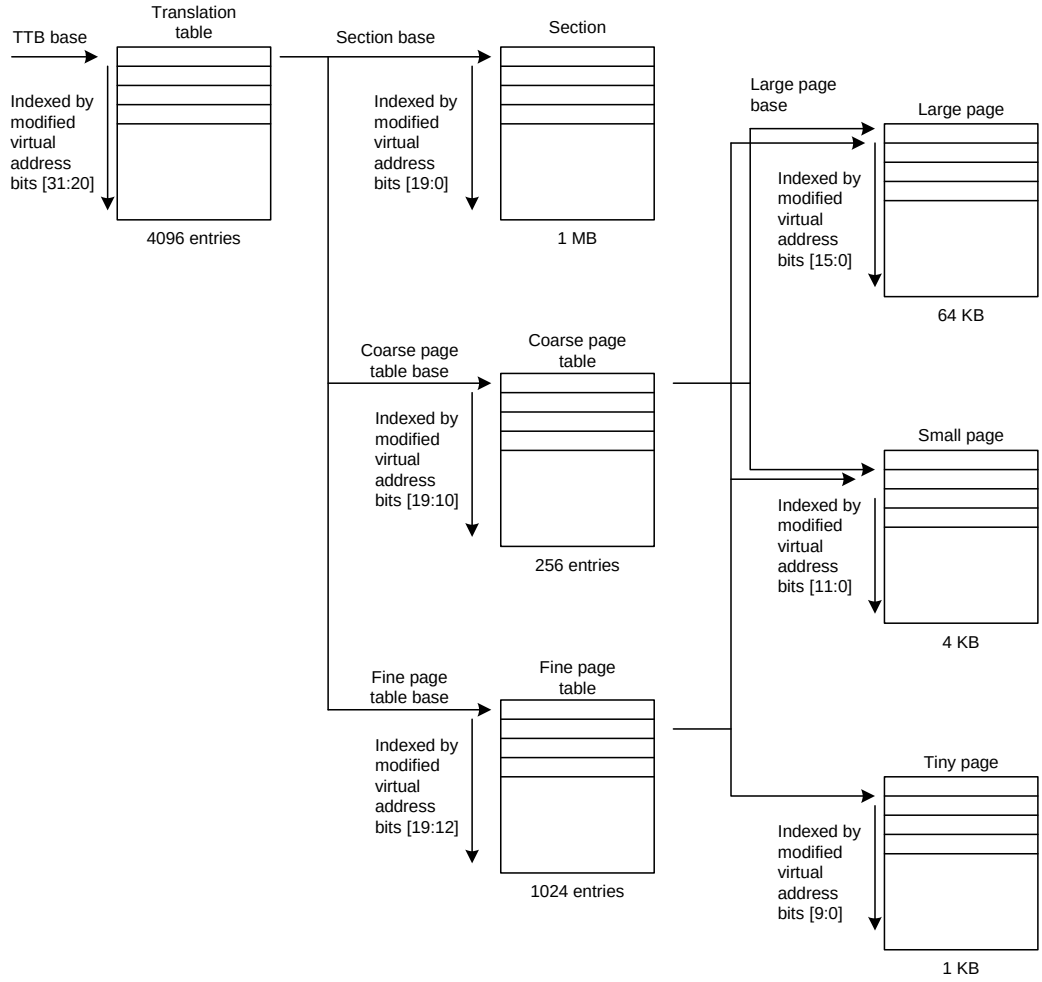


Figure 21: Translating page tables

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 22 shows the concatenation and address:

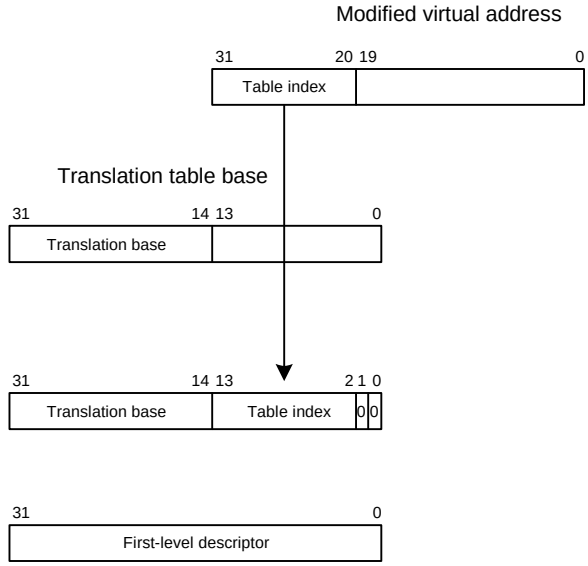


Figure 22: Accessing translation table first-level descriptors

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 23 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.

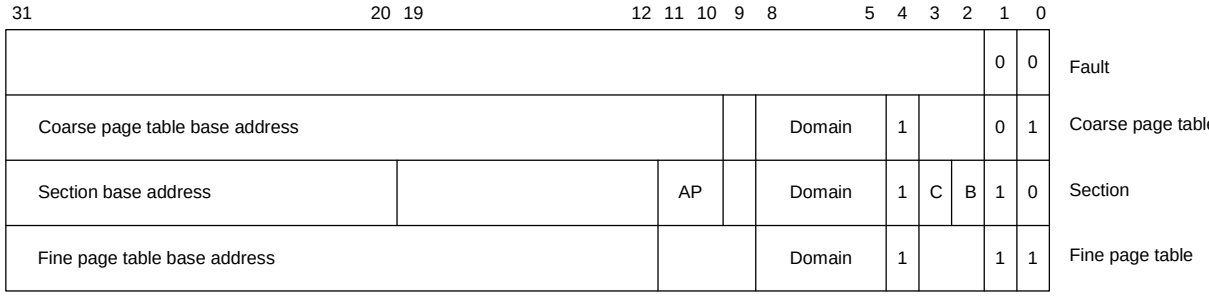


Figure 23: First-level descriptor

Table 31 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 98 and "Fault Address and Fault Status registers" on page 115 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.

Table 31: Priority encoding of fault status

Bits			
Section	**Coarse**	**Fine**	**Description**
[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
[1:0]	[1:0]	[1:0]	These bits indicate the page size and validity, and are interpreted as shown in Table 32, “Interpreting first-level descriptor bits [1:0],” on page 105.

Table 31: 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 32: Interpreting first-level descriptor bits [1:0]

Section descriptor

A section descriptor provides the base address of a 1 MB block of memory. Figure 24 shows the section descriptor format. Table 33 describes the section descriptor bits.

Figure 24: 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.
[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 33: 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 25 shows the coarse page table descriptor format; Table 34 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.

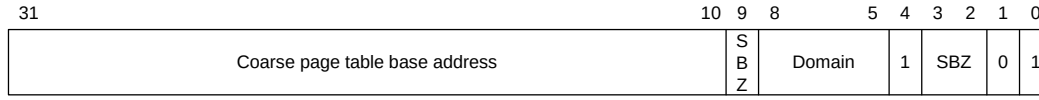


Figure 25: Coarse page table descriptor

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).

Table 34: Coarse page table descriptor bits

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

Table 34: 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 26 shows the format of a fine page table descriptor. Table 35 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.

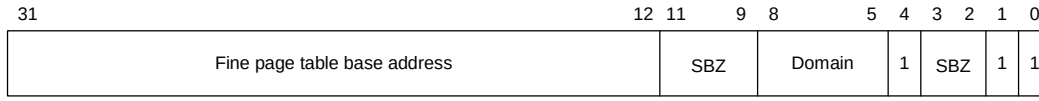


Figure 26: Fine page table descriptor

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.

Table 35: Fine page table descriptor bits

Bits	Description
[1:0]	Must be *11* to indicate a fine page table descriptor.

Table 35: Fine page table descriptor bits

Translating section references

Figure 27 shows the complete section translation sequence.

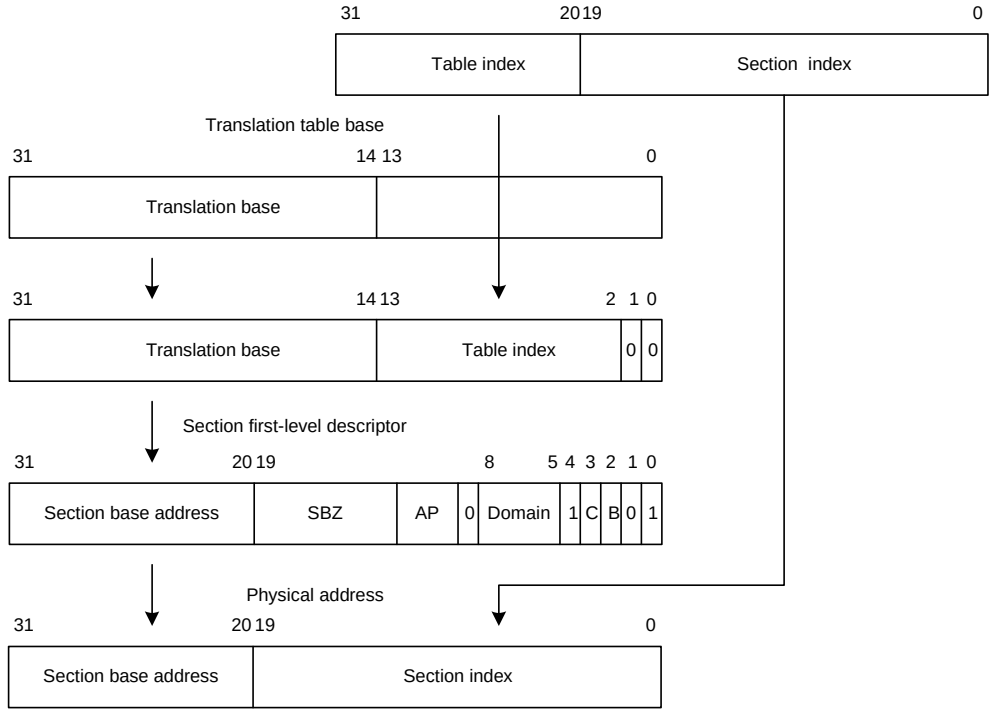


Figure 27: Section translation

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.

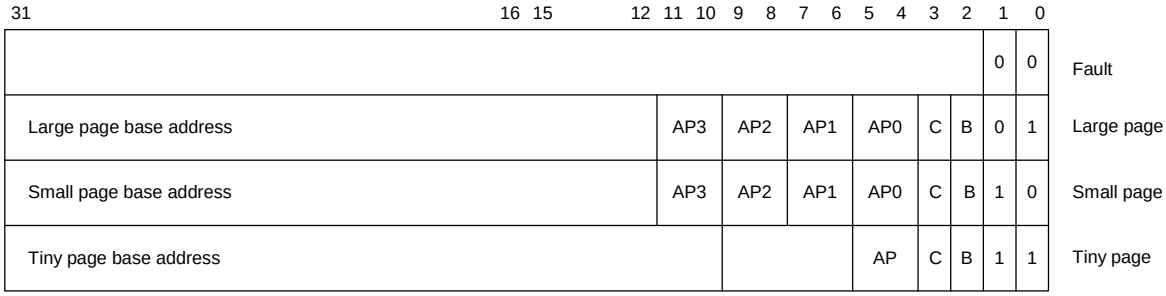


Figure 28: Second-level descriptor

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 36 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 36: Second-level descriptor bits

Bits			
Large	**Small**	**Tiny**	**Description**
[11:4]	[11:4]	[5:4]	Access permission bits. See "Domain access control" on page 117 and "Fault checking sequence" on page 118 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 37, “Interpreting page table entry bits [1:0],” on page 110.

Table 36: Second-level descriptor bits

The two least significant bits of the second-level descriptor indicate the descriptor type, as shown in this table.

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 37: 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 29 shows the complete translation sequence for a 64 KB large page.

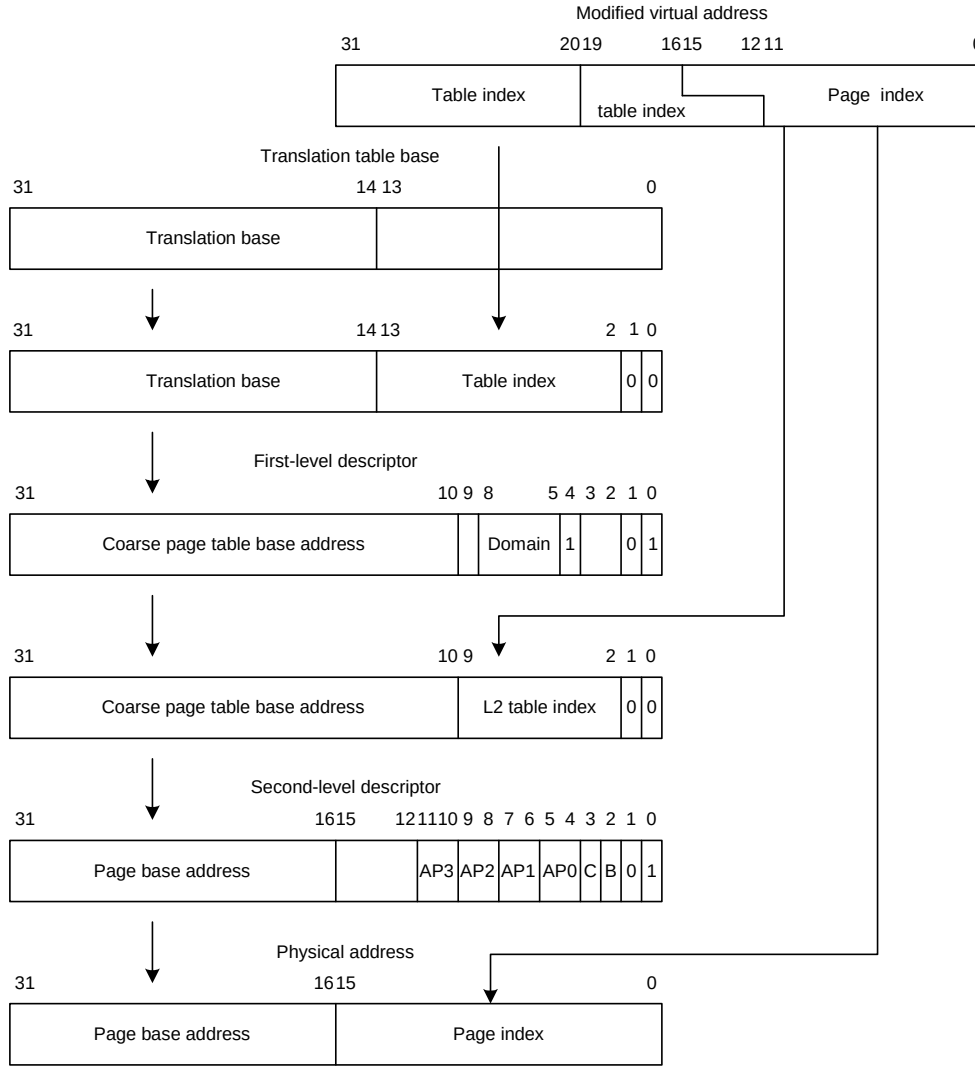


Figure 29: Large page translation from a coarse page table

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

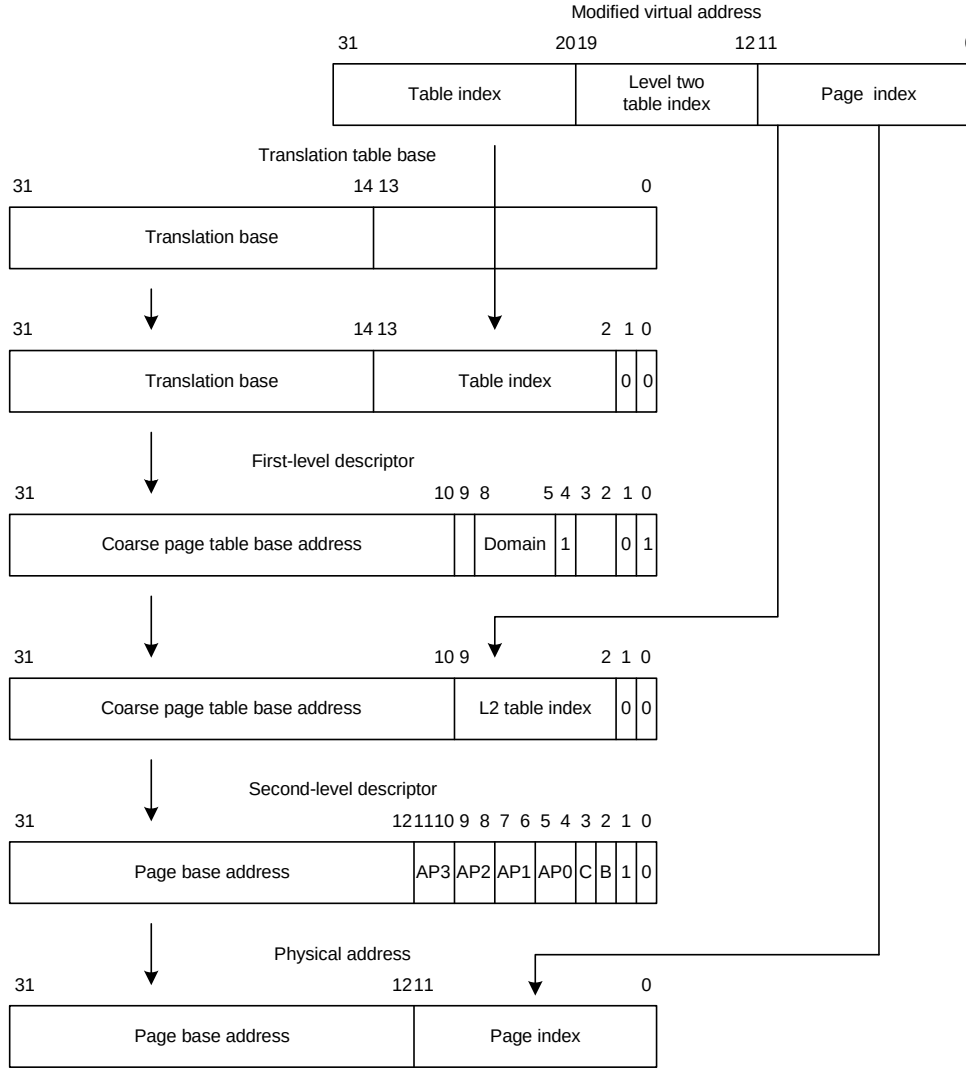


Figure 30: Small page translation from a coarse page table

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

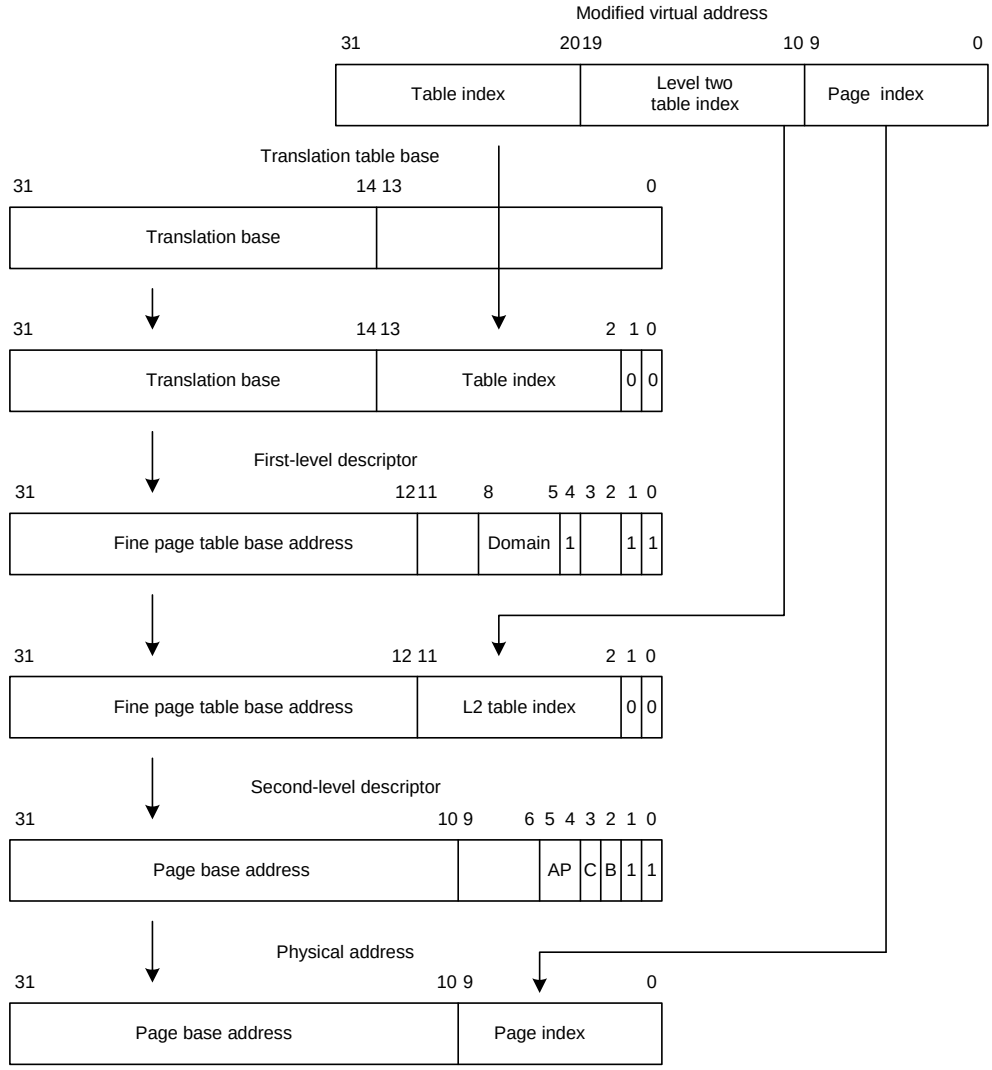


Figure 31: Tiny page translation from a fine page table

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 117 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 115).

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 38. 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	0b1100	Invalid
		Second level	0b1110	Valid
	Translation	Section page	0b0101	Invalid
			0b0111	Valid
	Domain	Section page	0b1001	Valid
			0b1011	Valid
	Permission	Section page	0b1101	Valid
			0b1111	Valid
Lowest	External abort	Section page	0b1000	Valid
			0b1010	Valid

Table 38: 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 39 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 39: 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 11, "R3: Domain Access Control register," on page 81 shows how the 32 bits of the register are allocated to define the 16 two-bit domains.

This table 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 40: Domain Access Control register, access control bits

This table 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 78).

AP	S	R	Privileged permissions	User permissions
0 0	0	0	No access	No access
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 41: 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 32 shows the sequence for both types of access.

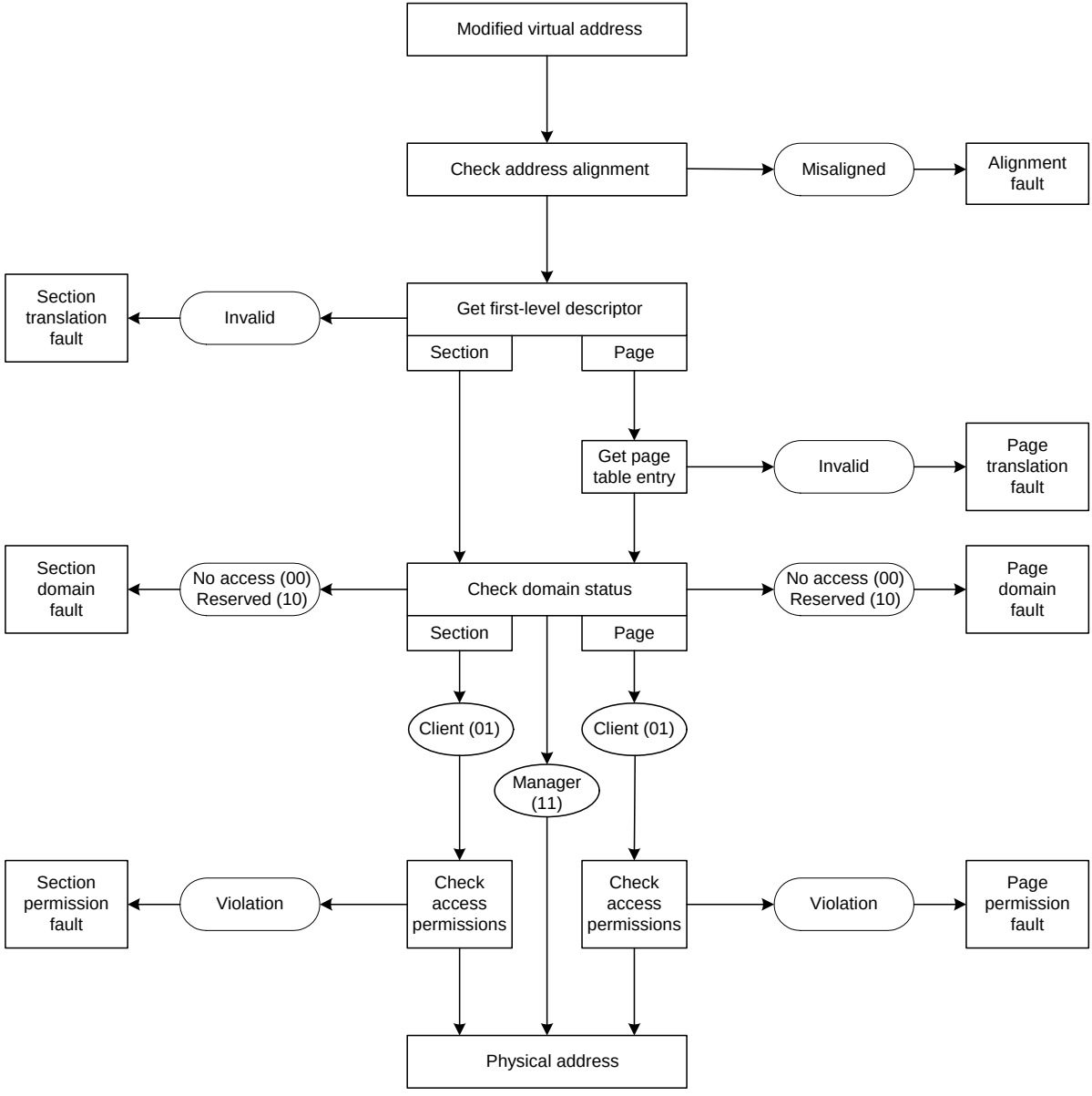


Figure 32: Sequence for checking faults

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 78), 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 41: "Interpreting access permission (AP) bits" on page 118. 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 41: "Interpreting access permission (AP) bits" on page 118. 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 41: "Interpreting access permission (AP) bits" on page 118. The interpretation depends on the setting of the S and R bits (see "R1: Control register," beginning on page 78). 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 41: "Interpreting access permission (AP) bits" on page 118).

  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, you must perform these steps:

1. Program the R2: Translation Table Base register and the R3: Domain Access Control register.
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 88).

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 92).

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 instruction cache is 8 KB in length, and the data cache is 4 KB in length.

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 42 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 43: "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 42: R1:Control register I and M bit settings for ICache

Table 43 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 43: Page table C bit settings for ICache

Table 44 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 44: 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 45: "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 44: R1: Control register I and M bit settings for DCache

Table 45 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 45: 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 33 shows a generic, virtually indexed, virtually addressed cache.

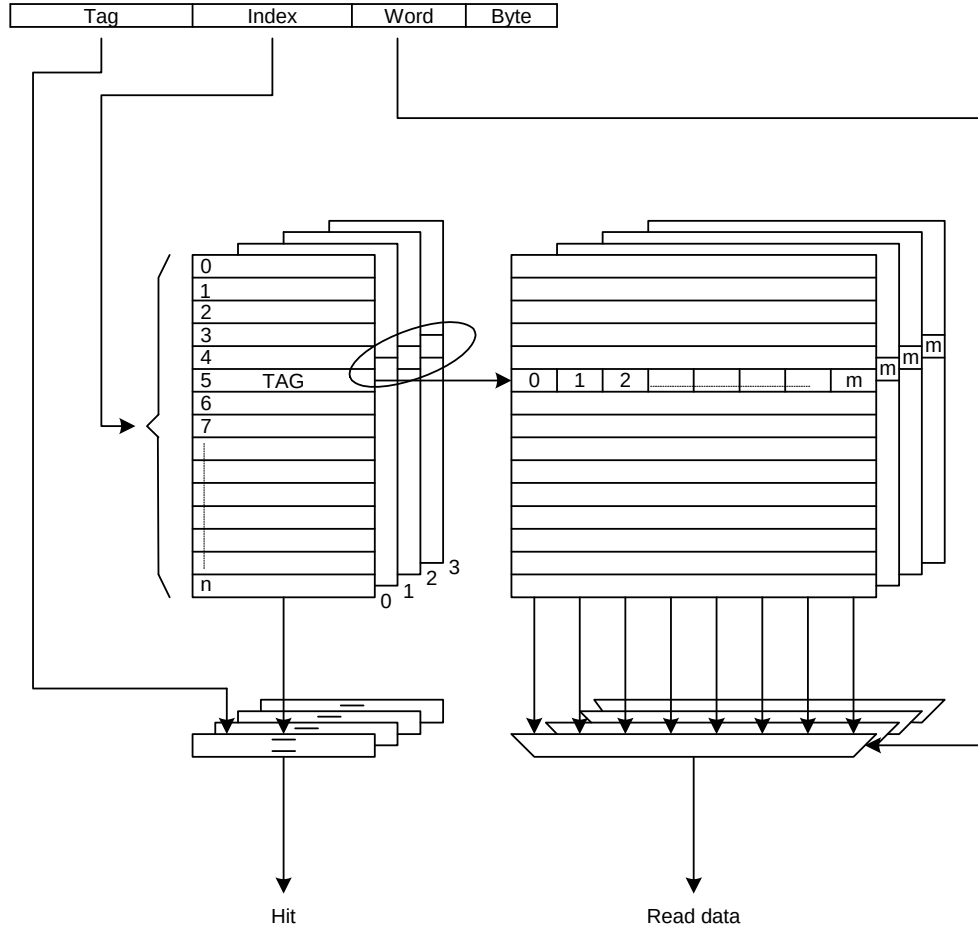


Figure 33: Generic virtually indexed, virtually addressed cache

Figure 34 shows the ARM926EJ-S cache format.

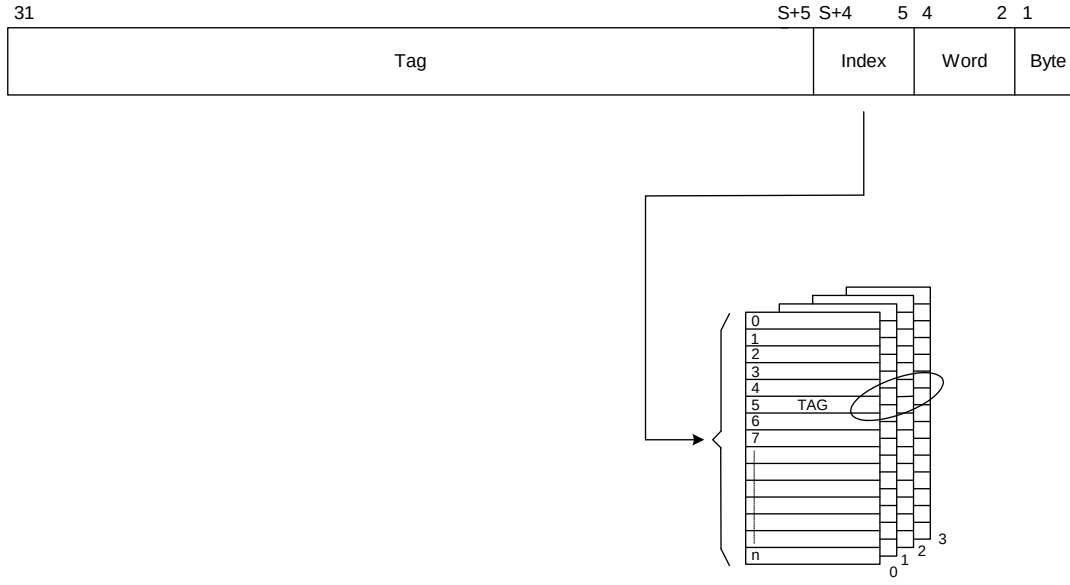


Figure 34: ARM926EJ-S cache associativity

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 46 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 46: Values of S and NSETS

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

Table 46: Values of S and NSETS

Figure 35 shows the set/way/word format for ARM926EJ-S caches.

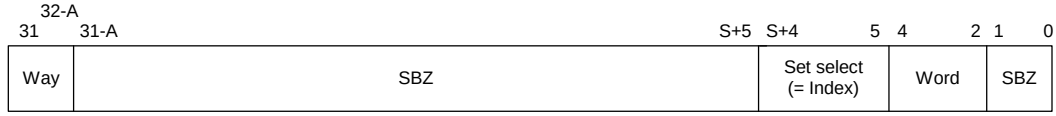


Figure 35: ARM926EJ-S cache set/way/word format

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 132.

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 84).

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 131).

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)
```

System Control Module

CHAPTER 4

The System Control Module configures and oversees system operations for the NS9360, and defines both the NS9360 AMBA High-speed Bus (AHB) arbiter system and system memory address space.

System Control Module features

The System Control Module uses the following to configure and maintain NS9360 system operations:

- AHB arbiter system
- System-level address decoding
- 10 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)
  - 8 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 140 for information about configuring the AHB arbiter.

The NS9360 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 36 shows an overview of the NS9360 high-speed bus architecture.

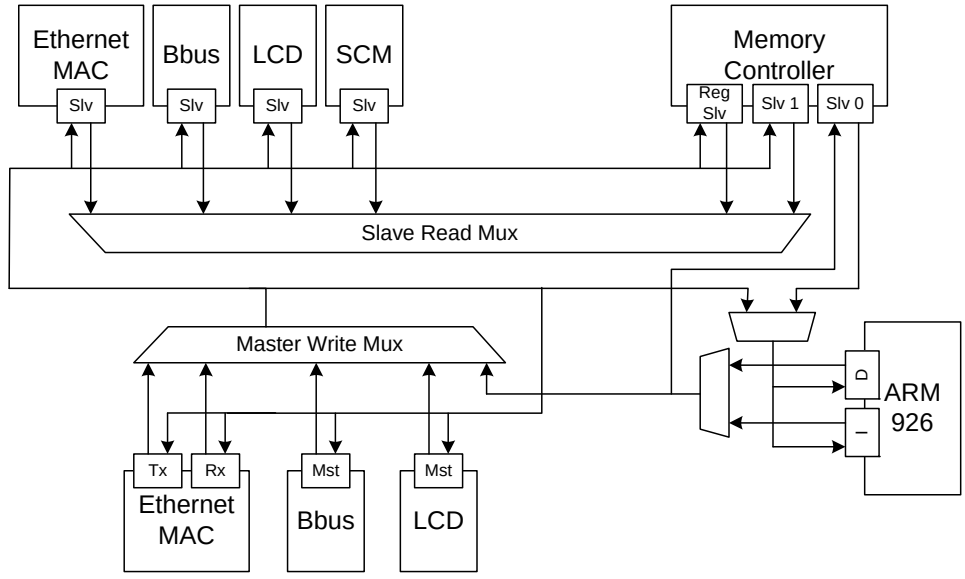


Figure 36: NS9360 bus architecture

The NS9360 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 BRR at every bus clock until one or more bus requests are registered.
2. The arbiter stops evaluating the BRR 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. 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:

- 4 AHB masters — Ethernet Rx, Ethernet Tx, BBus, and LCD.
- Memory clock frequency — 90 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:

= [(90 MHz/2) / (16 clock cycles per access x 4 masters)] x 32 bytes

= 22.5 Mbps/master

The factor 100 MHz/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 90 Mbps total to be split by all 5 masters.

Example 1

Since the 22.5 Mbps/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%, BBus
BRC1[15:8]	= 8'b1_0_00_0000	channel enabled, 100%, ARM926EJ-S
BRC1[7:0]	= 8'b1_0_00_0101	channel enabled, 100%, LCD
BRC2[31:24]	= 8'b0_0_00_0000	channel disabled
BRC2[23:16]	= 8'b0_0_00_0000	channel disabled
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 22.5 Mbps and the other masters need less than 22.5 Mbps. These are the new requirements:

- Ethernet Rx — 12.5 Mbps
- Ethernet Tx — 12.5 Mbps
- BBus — 4 Mbps
- LCD — 25 Mbps
- Total — 54 Mbps

This configuration is possible because the total bandwidth is less than the 90 Mbps available. The LCD master will be configured to have two arbiter slots, resulting in a total of 5 masters.

The available bandwidth per master is calculated using this formula:

Bandwidth per master:

= [(90 MHz/2) / (16 clock cycles per access x 5 masters)] x 32 bytes

= 18 Mbps/master

If the LCD is configured for two arbiter channel slots, then, there are 36 Mbps available, which is greater than the 25 Mbps required. Each of the other masters have 18 Mbps 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_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 second slot
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

Note that the BBus requires 4 Mbps but has been allocated 18 Mbps. The BBus bandwidth can be reduced using the bandwidth reduction field (in the BRC registers). To reduce the available bandwidth to 25%, to 4.5 Mbps 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 47 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 183 through "System Memory Chip Select 3 Dynamic Memory Base and Mask registers" on page 186 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 0 Dynamic memory (default)
0x1000 0000 – 0x1FFF FFFF	256 MB	System memory chip select 1 Dynamic memory (default)
0x2000 0000 – 0x2FFF FFFF	256 MB	System memory chip select 2 Dynamic memory (default)
0x3000 0000 – 0x3FFF FFFF	256 MB	System memory chip select 3 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	Reserved
0x9000 0000 – 0x9FFF FFFF	256 MB	BBus peripherals
0xA000 0000 – 0xA03F FFFF	4 MB	Reserved
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

Table 47: System address map

Address range	Size	System functions
0xA080 0000 – 0xA08F FFFF	1 MB	LCD controller
0xA090 0000 – 0xA09F FFFF	1 MB	System Control Module
0xA0A0 0000 – 0xFFFF FFFF	1526	Reserved

Table 47: System address map

Table 48 shows the hmaster[3:0] assignments for NS9360.

Master Name	hmaster[3:0] assignment
ARM926 I/D	0000
Ethernet Rx	0001
Ethernet Tx	0010
Reserved	0011
Reserved	0100
BBus	0101
LCD	0110

Table 48: Hmaster encoding

Programmable timers

NS9360 provides 10 programmable timers:

- Software watchdog timer
- 8 general purpose timers
- Bus monitor timer

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

Eight 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 50 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 0–7 Control registers" on page 165).

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 164), 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. Disable the timer when the ARM926EJS is halted in debug mode.
- PWM function (described next)

PWM function

The timers/counters can be configured to provide a simple PWM function. Each PWM function requires concatenating two timers/counters together, resulting in four PWM outputs. One timer/counter controls the pulse width and the other timer/counter controls the period.

The PWM outputs are fixed as shown:

- PWM0 — Timers 0 and 1
- PWM1 — Timers 2 and 3
- PWM2 — Timers 4 and 5
- PWM3 — Timers 6 and 7

Figure 37 illustrates the PWM function.

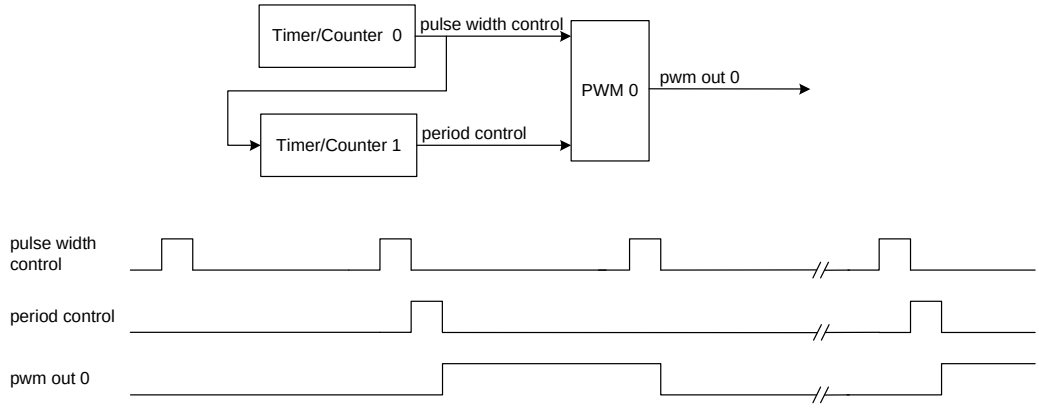


Figure 37: PWM function block diagram

As an alternative, you can use the timers independently rather than concatenating them. Follow these steps:

1. Start the timer1 with interval equal to period.
2. Start the timer2 with same interval as timer1, but after a required delay (period-pulse width nanoseconds).

This procedure allows 1% resolution. Be aware that the starting time of the second timer is critical. One way to ensure there are no problems is to disable the interrupts between the timer1 start time and the timer2 start time. Note also that there may be accuracy limitations with a smaller period.

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 NS9360 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 NS9360, and are managed using the Interrupt Config registers (see "Int (Interrupt) Config (Configuration) registers (0–31)" on page 169). 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 38 illustrates a 32-vector interrupt controller.

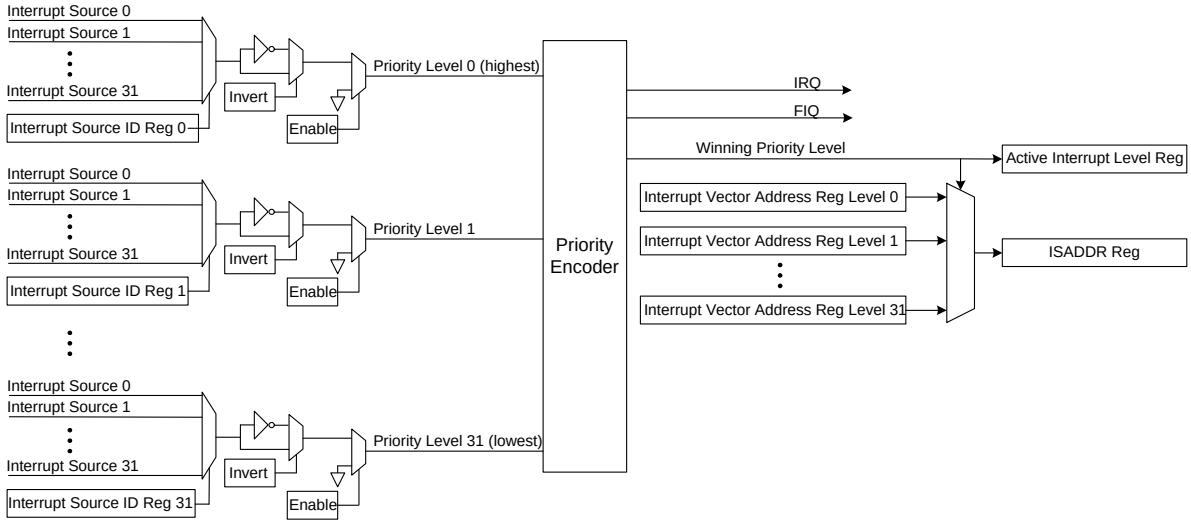


Figure 38: Interrupt controller block diagram

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 170). 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 NS9360 interrupt sources are assigned as shown:

Interrupt ID	Interrupt source
0	Watchdog Timer
1	AHB Bus Error
2	BBus Bridge Aggregate Interrupt
3	Reserved
4	Ethernet Module Receive Interrupt
5	Ethernet Module Transmit Interrupt
6	Ethernet Phy Interrupt
7	LCD Module interrupt
8	Serial Port B Receive Interrupt
9	Serial Port B Transmit Interrupt
10	Serial Port A Receive Interrupt
11	Serial Port A Transmit Interrupt
12	Serial Port C Receive Interrupt
13	Serial Port C Transmit Interrupt
14	I^2C 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	RTC Interrupt
25	USB Host Interrupt

Interrupt ID	Interrupt source
26	USB Device Interrupt
27	IEEE 1284 Interrupt
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 NS9360 system attributes:

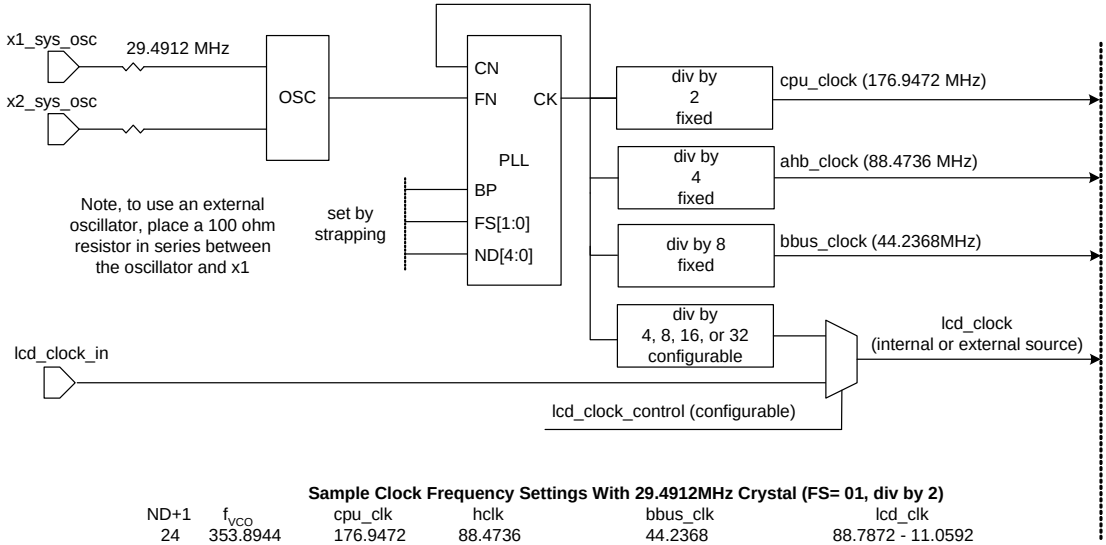
- Little endian/big endian mode
- Watchdog timer enable
- Watchdog timeout generates IRQ/FIQ/RESET
- Watchdog timeout interval
- Enable/disable ERROR response for misaligned data access
- System module clock enables
- Enable access to internal registers in USER mode
- Bus monitor enable
- Bus monitor timeout interval
- Bus 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 181). 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 181). Once the new settings have been written, the PLL SW change bit must be set. 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. NS9360 provides internal pullups to produce a default configuration; see "Bootstrap initialization" on page 153 for information about the powerup configuration.

Figure 39 shows how the PLL clock is used to provide the NS9360 system clocks.



Sample Clock Frequency Settings With 29.4912MHz Crystal (FS= 01, div by 2)

ND+1	f_{VCO}	cpu_clk	hclk	bbus_clk	lcd_clk
24	353.8944	176.9472	88.4736	44.2368	88.7872 - 11.0592
23	339.1488	169.5744	84.7872	42.3936	84.7872 - 10.5984
22	324.4032	162.2016	81.1008	40.5504	81.1008 - 10.1376
21	309.6576	154.8288	77.4144	38.7072	77.4144 - 9.6768
20	294.9120	147.4560	73.7280	36.8640	73.7280 - 9.2160
19	280.1644	140.0832	70.0416	35.0208	70.0416 - 8.7552
18	265.4208	132.7104	66.3552	33.1776	66.3552 - 8.2944
17	250.6752	125.3376	62.6688	31.3344	62.6688 - 7.8336
16	235.9296	117.9648	58.9824	29.4912	58.9824 - 7.3728
15	221.1840	110.5920	55.2960	27.6480	55.2960 - 6.9120
14	206.4384	103.2192	51.6096	24.8048	51.6096 - 6.4512

Figure 39: NS9360 system clock generation (PLL)

You can use this formula to calculate the system clock frequencies if a different system oscillator frequency is used:

f_{vco} = (f_{osc} x (ND + 1) / FS)

f_{cpu_clk} = f_{vco} / 2

f_{hclk} = f_{vco} / 4

f_{bbuys_clk} = f_{vco} / 8

f_{lcd_clk} = programmable, f_{vco} / 4, 8, 16, or 32

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 49 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 50 shows PLL ND[4:0] multiplier values. The NS9360 BGA layout in Chapter 19, "Packaging," shows bootstrapping pins.

Pin name	Configuration bits
rtck_out	**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)
gpio[24] gpio[20]	**Chip select 1 data width bootstrap select** 00 16 bits 01 8 bits 11 32 bits
gpio[49]	**Chip select polarity** 0 Active high 1 Active low
gpio[44]	**Endian mode** 0 Big endian 1 Little endian
reset_done	**Bootup mode** 0 Boot from SDRAM using serial SPI EEPROM 1 Boot from flash/ROM
gpio[19]	Reserved: This pin must not be pulled to a logic 0 until reset_done is a logic 1.
gpio[17], gpio[12], gpio[10], gpio[8], gpio[4]	**PLL ND[4:0] (PLL multiplier, ND+1)** See Table 50: "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 49: 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**
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**

Table 50: PLL ND[4:0] multiplier values

Register configuration: gpio 17, 12, 10, 8, 4	Multiplier
00011	**7**
00000	**6**
00001	**5**
00110	**4**
00111	**3**
00100	**2**
00101	**1**

Table 50: PLL ND[4:0] multiplier values

Sample frequency settings with 29.4912 MHz crystal

These are sample frequency settings for each speed grade:

- 176.9472 MHz: pulldown gpio[12], gpio[10], gpio[4]
- 154.8288 MHz: pulldown gpio[12], gpio[10], gpio[8]
- 103.2192 MHz: pulldown gpio[17], gpio[10], gpio[8], gpio[4]

Additional GPIO pins

There are 32 additional GPIO pins that are used to create a general purpose, user-defined ID register (see "Gen ID register" on page 192). 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 51 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			

Table 51: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 0014–A090 0040		Reserved		
A090 0044	Timer 0 Reload Count register			
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 – A090 0080		Reserved		
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 – A090 A090 00C0		Reserved		
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			

Table 51: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
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			
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

Table 51: System Control module registers

Offset	[31:24]	[23:16]	[15:8]	[7:0]
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			
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 – A090 01CC	Reserved			
A090 01D0	System Memory Chip Select 0 Dynamic Memory Base			
A090 01D4	System Memory Chip Select 0 Dynamic Memory Mask			

Table 51: System Control module registers

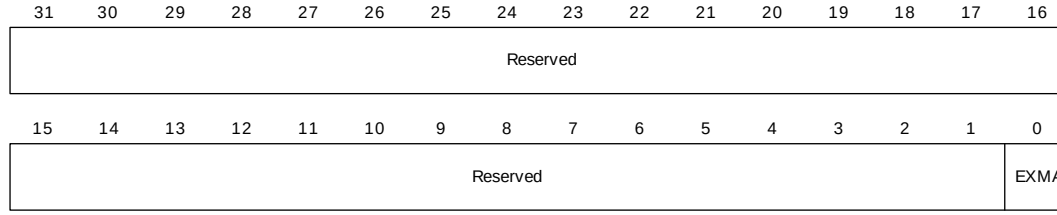
Offset	[31:24]	[23:16]	[15:8]	[7:0]
A090 01D8	System Memory Chip Select 1 Dynamic Memory Base			
A090 01DC	System Memory Chip Select 1 Dynamic Memory Mask			
A090 01E0	System Memory Chip Select 2 Dynamic Memory Base			
A090 01E4	System Memory Chip Select 2 Dynamic Memory Mask			
A090 01E8	System Memory 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			
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			
A090 0224	RTC Clock Control			

Table 51: 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.



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 52: AHB Arbiter Gen Configuration register

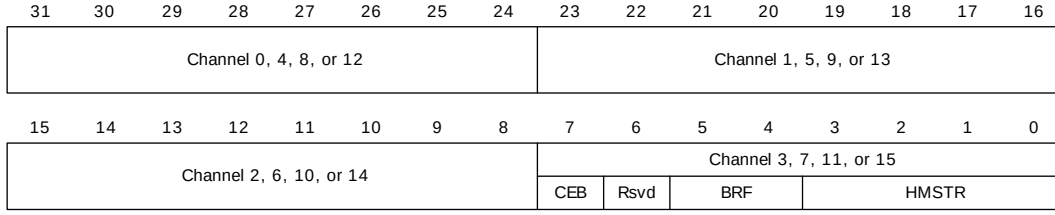
BRC0, BRC1, BRC2, and BRC3 registers

Address: A090 0004 / 0008 / 000C / 0010

The BRC[0:3] registers control the AHB arbiter bandwidth allocation scheme. Table 53 shows how the channels are assigned in the four registers. Table 54 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 53: BRC channel assignment



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D07	R/W	CEB	0x1	**Channel enable bit** 0 Disable 1 Enable
D06	N/A	Reserved	N/A	N/A

Table 54: BRC0, BRC1, BRC2, BRC3 register

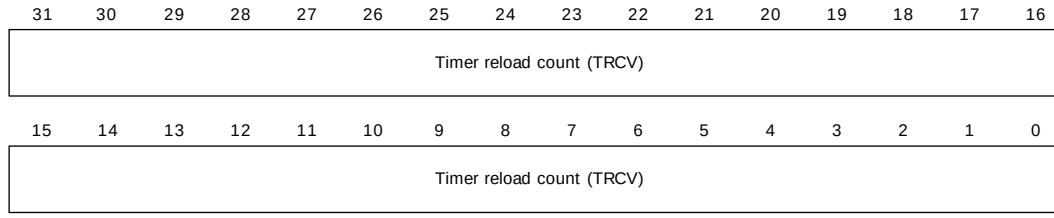
Bits	Access	Mnemonic	Reset	Description
D05:04	R/W	BRF	0x0	**Bandwidth reduction field** 00 100% 01 75% 10 50% 11 25% Programs the weight for each AHB bus master. Used to limit the round robin scheduler.
D03:00	R/W	HMSTR	n (where n is the actual HMSTR #)	**hmaster** Program a particular AHB bus master number here. Note that a particular master can be programmed to more than one channel.

Table 54: BRC0, BRC1, BRC2, BRC3 register

Timer 0–7 Reload Count registers

Address: A090 0044 (Timer 0) / 0048 / 004C / 0050 / 0054 / 0058 / 005C / 0060 (Timer 7)

The Timer Reload registers hold the up/down reload value.



Register bit assignment

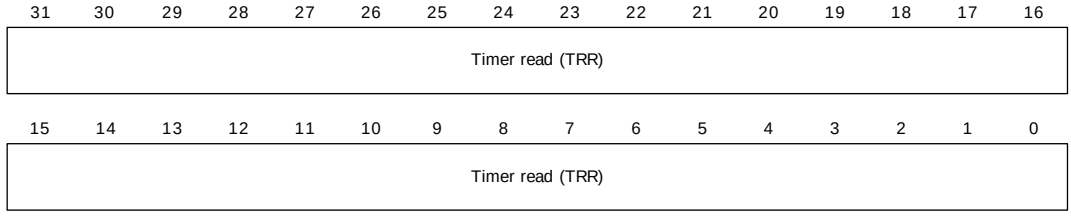
Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	TRCV	0x0	**Timer Reload Count register value** Value loaded into the Timer register after the timer is enabled and after the terminal count has been reached, if the reload enable bit in the corresponding Timer Control register is set.

Table 55: Timer Reload Count register

Timer 0–7 Read register

Address: A090 0084 (Timer 0) / 0088 / 008C / 0090 / 0094 / 0098 / 009C / 00A0 (Timer 7)

The Timer Read registers read the current state of each Timer register.



Register bit assignment

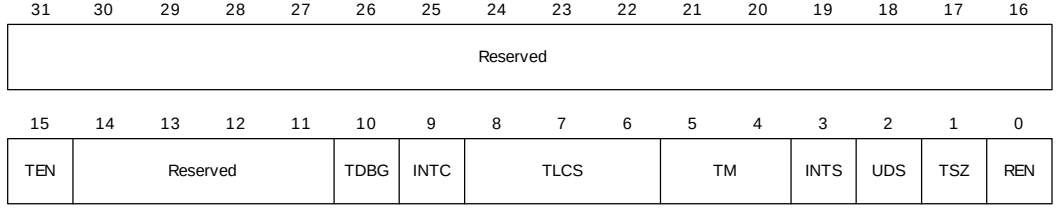
Bits	Access	Mnemonic	Reset	Description
D31:00	R	TRR	0x0	**Timer Read register** Reads the current state of each counter in a register.

Table 56: Timer Read register

Timer 0–7 Control registers

Address: A090 0190 (Timer 0) / 0194 / 0198 / 019C / 01A0 / 01A4 / 01A8 / 01AC (Timer 7)

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
D14:11	N/A	Reserved	N/A	N/A
D10	R/W	TDBG	0x0	**CPU debug mode** 0 Timer continues to run when CPU is halted in debug mode 1 Timer stops when CPU is halted in debug mode
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 ■ 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.

Table 57: Timer Control registers

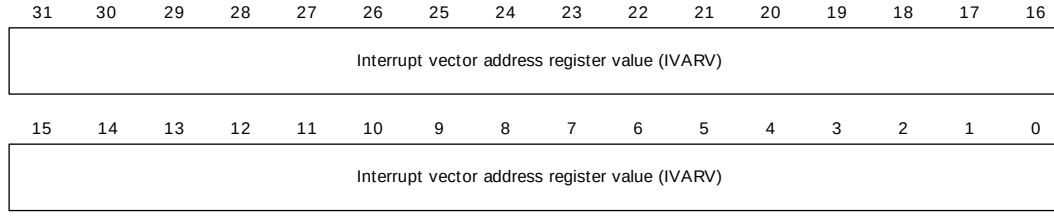
Bits	Access	Mnemonic	Reset	Description
D05:04	R/W	TM	0x0	**Timer mode** 00 Internal timer or external event 01 External low-level, gated timer 10 External high-level, gated timer 11 Concatenate the lower timer. Not applicable on timer 0. **Note:** When either external gated timer option is selected, 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 57: Timer Control registers

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.



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 58: Interrupt vector address register

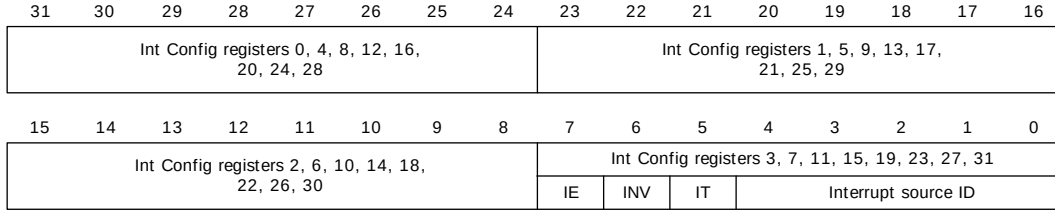
Int (Interrupt) Config (Configuration) registers (0–31)

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 60 shows how the 32 individual 8-byte registers are mapped in the eight 32-bit registers. Table 61 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 59: Interrupt configuration register address mapping



Register bit assignment

BIts	Access	Mnemonic	Reset	Definition
D07	R/W	IE	0x0	**Interrupt enable** 0 Interrupt is disabled 1 Interrupt is enabled

Table 60: Int Config register

BIts	Access	Mnemonic	Reset	Definition
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 **Note:** If FIQ is programmed, *Interrupt* must be the highest priority.
D04:00	R/W	ISD	0x0–0x1F	**Interrupt source ID** Assign an interrupt ID to each priority level. See "Interrupt sources," beginning on page 149, for the list of interrupt ID numbers.

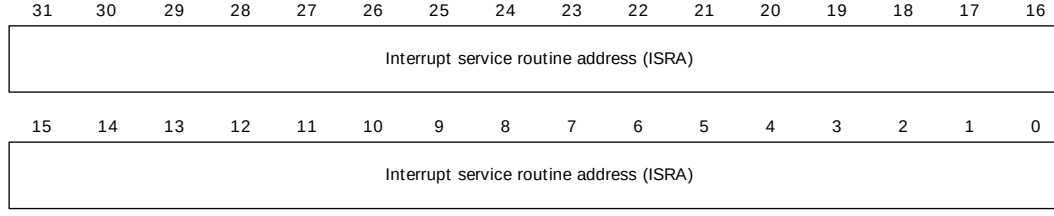
Table 60: Int Config register

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.



Register bit assignment

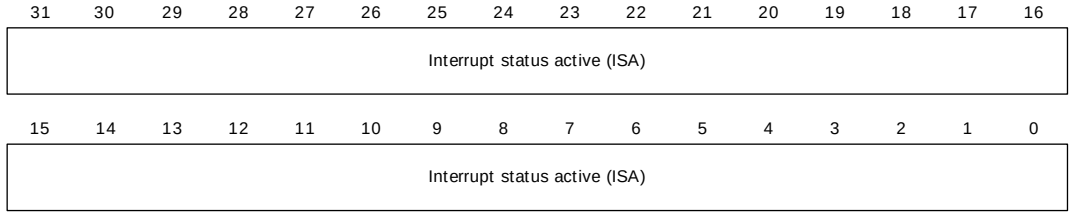
Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	ISRA	0x0	**Interrupt service routine address** ■ A read to this register updates the priority logic block, and masks the current and any lower priority interrupt requests. ■ A write of any value to this register clears the mask, to allow lower priority interrupts to become active.

Table 61: ISRADDR register

Interrupt Status Active

Address: A090 0168

The Interrupt Status Active register shows the current interrupt request.



Register bit assignment

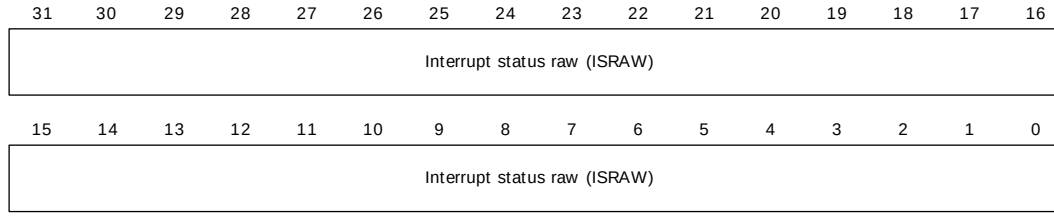
Bits	Access	Mnemonic	Reset	Description
D31:00	R	ISA	0x0	**Interrupt status active** Provides the status of all active, enabled interrupt request levels, where bit 0 is for the interrupt assigned to level 0, bit 1 is for the interrupt assigned to level 1, and so on through bit 31 for the interrupt assigned to level 31.

Table 62: Interrupt Status Active register

Interrupt Status Raw

Address: A090 016C

The Interrupt Status Raw register shows all current interrupt requests.



Register bit assignment

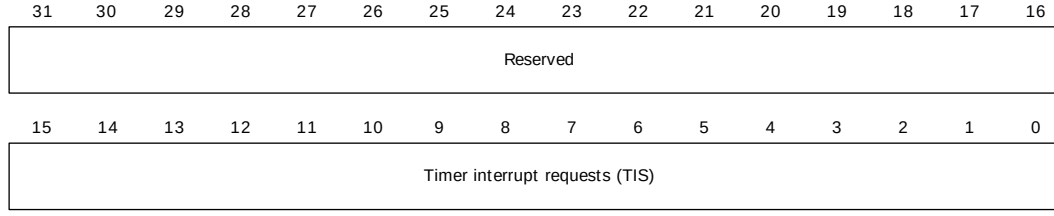
Bits	Access	Mnemonic	Reset	Description
D31:00	R	ISRAW	0x0	**Interrupt status raw** Provides the status of all active, enabled, and disabled interrupt request levels, where bit 0 is for the interrupt assigned to level 0, bit 1 is for the interrupt assigned to level 1, and so on through bit 31 for the interrupt assigned to level 31.

Table 63: 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 64: 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
D31:09	N/A	Reserved	N/A	N/A
D08	R/W	SDBG	0x0	**CPU debug mode** 0 Timer continues to run when CPU is halted in debug mode 1 Timer stops when CPU is halted in debug mode
D07	R/W	SWWE	0x0	**Software watchdog enable** 0 Software watchdog disabled 1 Software watchdog enabled. Once this is set, it cannot be cleared.

Table 65: Software Watchdog Configuration register

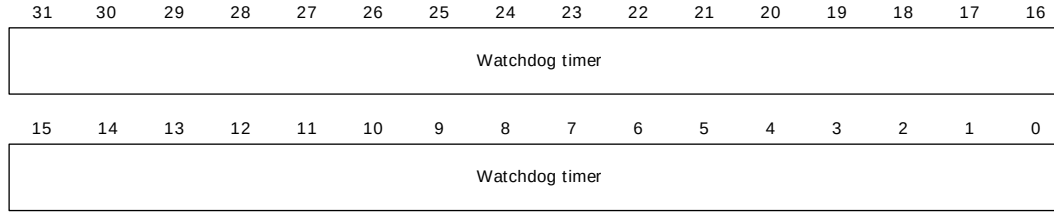
Bits	Access	Mnemonic	Reset	Description
D06	N/A	Reserved	N/A	N/A
D05	R/W	SWWI	0x0	**Software watchdog interrupt clear** Write a 1, then a 0 to this bit to clear the software watchdog interrupt.
D04	R/W	SWWIC	0x0	**Software watchdog interrupt response** 0 Generate an interrupt 1 Generate the reset
D03	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 65: Software Watchdog Configuration register

Software Watchdog Timer register

Address: A090 0178

The Software Watchdog Timer register services the watchdog timer.



Register bit assignment

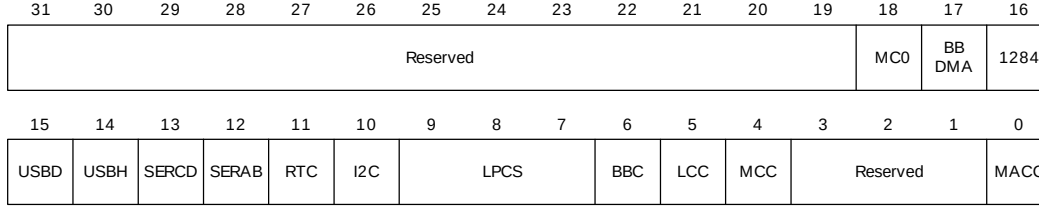
Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	WT	0x0	**Watchdog timer** ■ A *read* to this register gives the current value of the watchdog timer, but will not change the contents. ■ A *write* to the register changes the contents based on the write data value.

Table 66: 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:19	N/A	Reserved	N/A	N/A
D18	R/W	MC0	0	**Memory clock 0** 0 Clock enabled 1 Clock disabled
D17	R/W	BBDMA	1	**BBus DMA** 0 Clock disabled 1 Clock enabled

Table 67: Clock Configuration register

Bits	Access	Mnemonic	Reset	Description
D16	R/W	1284	1	**IEEE 1284** 0 Clock disabled 1 Clock enabled
D15	R/W	USBD	1	**USB device** 0 Clock disabled 1 Clock enabled
D14	R/W	USBH	1	**USB host** 0 Clock disabled 1 Clock enabled
D13	R/W	SERCD	1	**Serial C/D** 0 Clock disabled 1 Clock enabled
D12	R/W	SERAB	1	**Serial A/B** 0 Clock disabled 1 Clock enabled
D11	R/W	RTC	1	**Real time clock** 0 Clock disabled 1 Clock enabled
D10	R/W	I^2C	1	**I^2C** 0 Clock disabled 1 Clock enabled
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

Table 67: Clock Configuration register

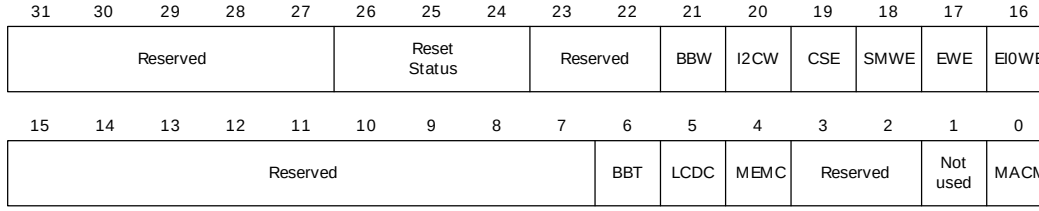
Bits	Access	Mnemonic	Reset	Description
D04	R/W	MCC	0x1	**Memory controller** 0 Clock disabled 1 Clock enabled
D03:01	N/A	Reserved	N/A	N/A
D00	R/W	MACC	0x1	**Ethernet MAC** 0 Clock disabled 1 Clock enabled

Table 67: 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Definition
D31:27	N/A	Reserved	N/A	N/A

Table 68: Reset and Sleep Control register

Bits	Access	Mnemonic	Reset	Definition
D26:24	R	RSTAT	N/A	**Reset status** Determines the cause of the last chip reset. 001 External reset using reset_n 010 External reset using sreset_n 011 PLL change reset 100 Software watchdog reset 101 AHB bus monitor reset
D23:22	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	I^2CW	0x0	**I^2C 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.
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	EI0WE	0x0	**Ext interrupt 0 wake-up enable** 0 Do not wake up on Ext interrupt 0 input signal. 1 Wake up on active low Ext interrupt 0 input signal.
D15:07	N/A	Reserved	N/A	N/A

Table 68: Reset and Sleep Control register

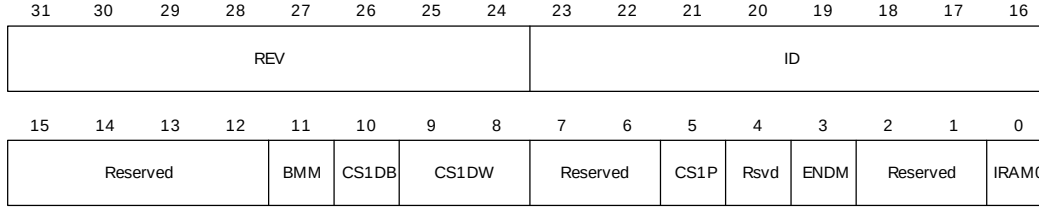
Bits	Access	Mnemonic	Reset	Definition
D06	R/W	BBT	0x1	**BBus top** 0 Module reset 1 Module enabled
D05	R/W	LCDC	0x1	**LCD controller** 0 Module reset 1 Module enabled
D04	R/W	MEMC	0x1	**Memory controller** 0 Module reset 1 Module enabled
D03:02	N/A	Reserved	N/A	N/A
D01	R/W	Not used	0x0	Must be written to 0.
D00	R/W	MACM	0x1	**Ethernet MAC** 0 Module reset 1 Module enabled

Table 68: 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R	REV	0x0	**Revision** Indicates the NS9360 hardware revision.
D23:16	R	ID	0x1	**ID** Indicates the NS9360 identification.
D15:12	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 rtck_out	**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.
D09:08	R	CS1DW	HW strap gpio[24], gpio[20]	**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.
D07:06	N/A	Reserved	N/A	N/A
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

Table 69: Miscellaneous System Configuration and Status register

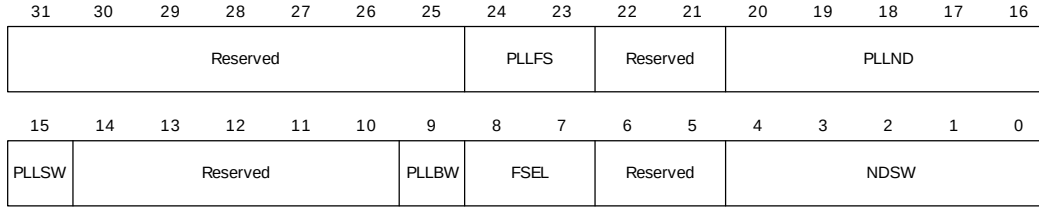
Bits	Access	Mnemonic	Reset	Description
D03	R/W	ENDM	HW strap gpio[44]	**Endian mode** 0 Little endian mode 1 Big endian mode
D02:01	N/A	Reserved	N/A	N/A
D00	R/W	IRAM0	0x1	**Internal register access mode bit 0** 0 Allow access to internal registers using PRIVILEGED mode only 1 Allow access to internal registers using PRIVILEGED or USER mode.

Table 69: Miscellaneous System Configuration and Status register

PLL Configuration register

Address: A090 0188

The PLL Configuration register configures the PLL.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:25	N/A	Reserved	N/A	N/A
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	N/A	Reserved	N/A	N/A

Table 70: PLL Configuration register

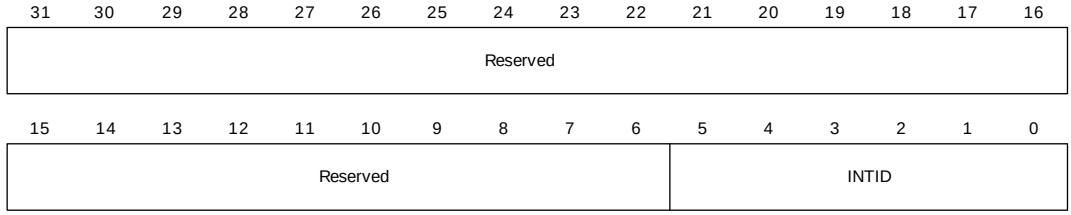
Bits	Access	Mnemonic	Reset	Description
D20:16	R	PLLND	HW strap gpio[17], gpio[12], gpio[10], gpio[8], gpio[4]	**PLL ND status[4:0]** Status register to determine the powerup strapping settings or the new settings as changed by software.
D15	W	PLLSW	0x0	**PLL SW change** Write a 1 to this bit to change the PLL settings as defined in bits D09:00. **Note**: The chip is reset after this bit is written to a 1 allowing the PLL to reset and lock to the new settings.
D14:10	N/A	Reserved	N/A	N/A
D09	R/W	PLLBW	0x0	**PLL bypass SW** Always set to 0.
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	N/A	Reserved	N/A	N/A
D04:00	R/W	NDSW	0x1A	**PLL ND SW [4:0]** PLL multiplier (ND+1).

Table 70: 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

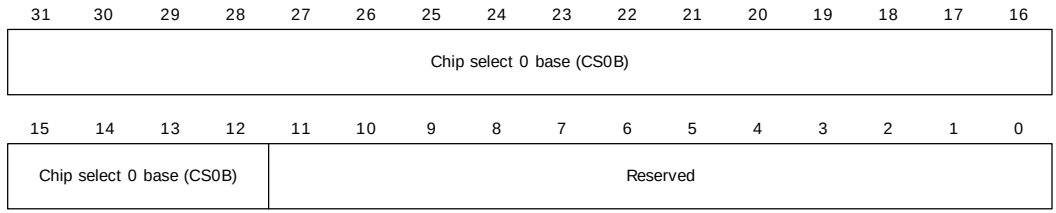
Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A
D05:00	R	INTID	0x0	**Interrupt** The level of the current active interrupt.

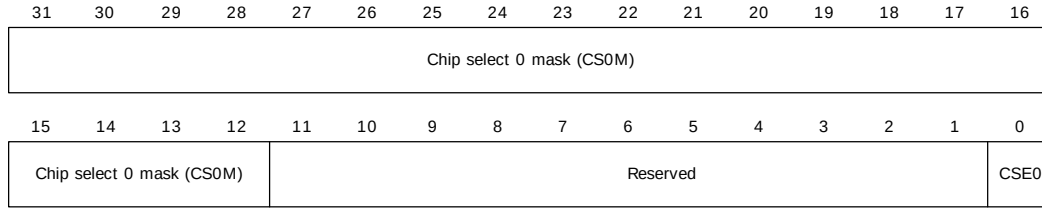
Table 71: Active Interrupt Level Status register

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 0, with a minimum size of 4K. The powerup default settings produce a memory range of 0x0000 0000 — 0x0FFF FFFF.





Register bit assignment

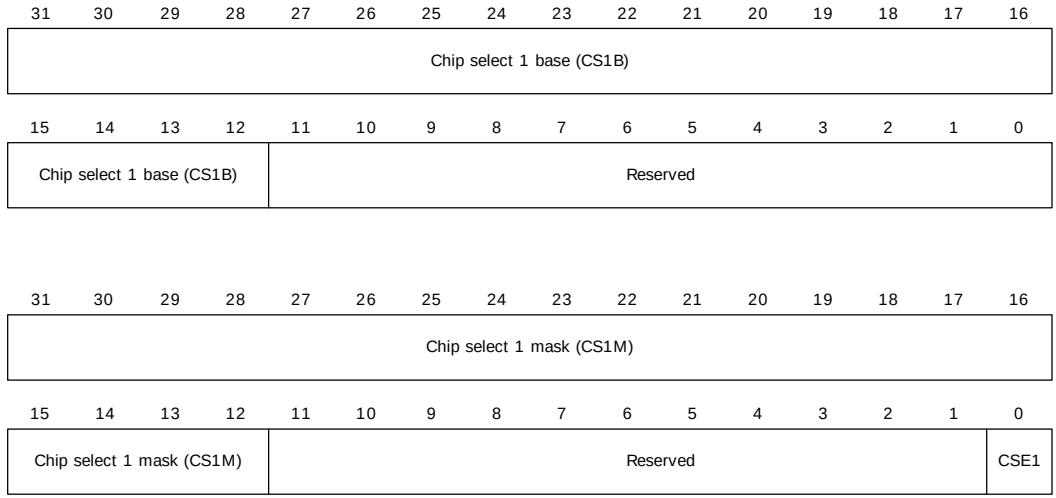
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS0B	0x00000	**Chip select 0 base** Base address for chip select 0
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS0M	0xF0000	**Chip select 0 mask** Mask or size for chip select 0
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE0	0x1	**Chip select 0 enable** 0 Disable chip select 1 Enable chip select

Table 72: 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.



Register bit assignment

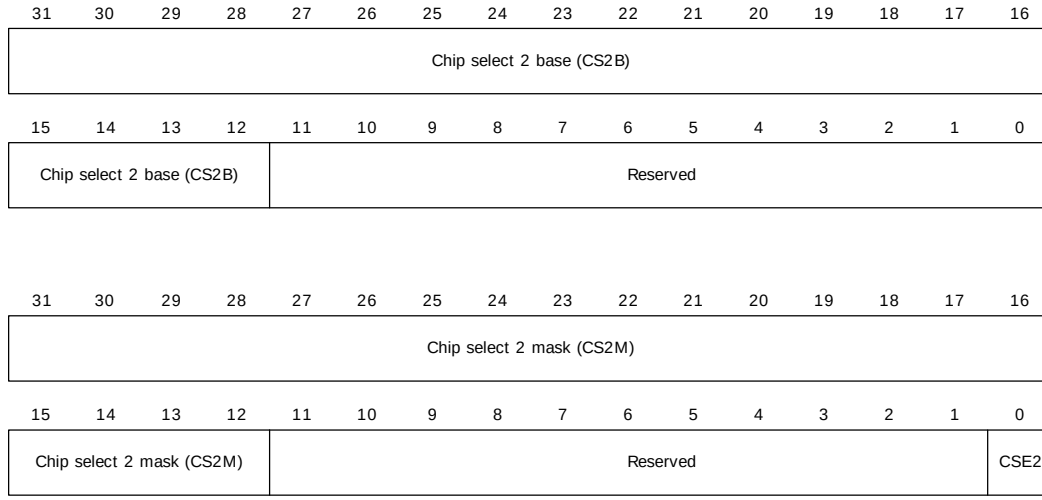
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS1B	0x10000	**Chip select 1 base** Base address for chip select 1
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS1M	0xF0000	**Chip select 1 mask** Mask or size for chip select 5
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE1	0x1	**Chip select 1 enable** 0 Disable chip select 1 Enable chip select

Table 73: 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 2, with a minimum size of 4K. The powerup default settings produce a memory range of 0x2000 0000 — 0x2FFF FFFF.



Register bit assignment

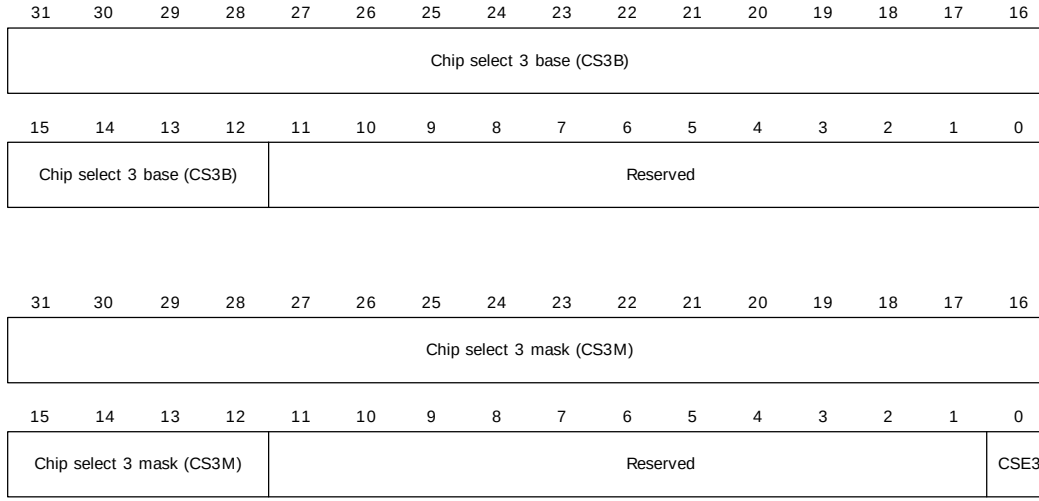
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS2B	0x20000	**Chip select 2 base** Base address for chip select 2
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS2M	0xF0000	**Chip select 2 mask** Mask or size for chip select 2
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE2	0x1	**Chip select 2 enable** 0 Disable chip select 1 Enable chip select

Table 74: 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 3, with a minimum size of 4K. The powerup default settings produce a memory range of 0x3000 0000 — 0x3FFF FFFF.



Register bit assignment

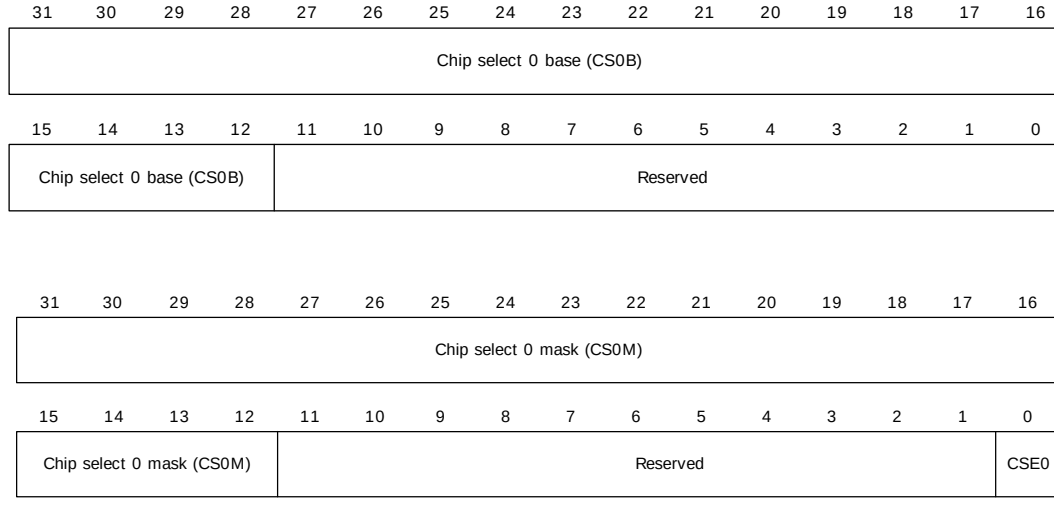
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS3B	0x30000	**Chip select 3 base** Base address for chip select 3
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS3M	0xF0000	**Chip select 3 mask** Mask or size for chip select 3
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE3	0x1	**Chip select 3 enable** 0 Disable chip select 1 Enable chip select

Table 75: 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.



Register bit assignment

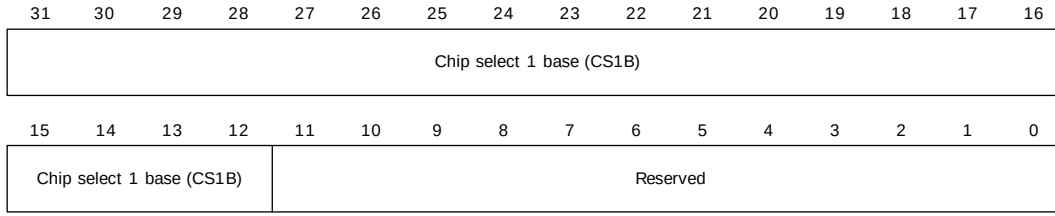
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS0B	0x40000	**Chip select 0 base** Base address for chip select 0.
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS0M	0xF0000	**Chip select 0 mask** Mask or size for chip select 0.
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE0	0x1	**Chip select 0 enable** 0 Disable chip select 1 Enable chip select

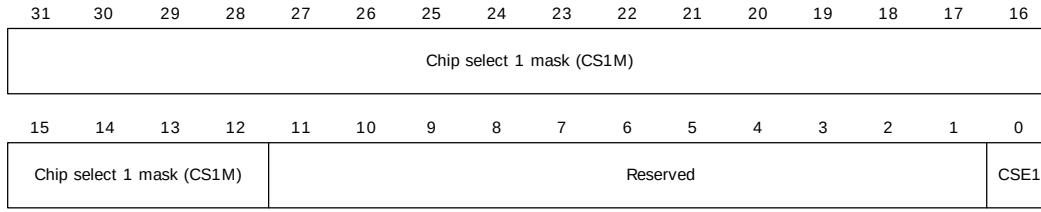
Table 76: System Memory Chip Select 0 Static Memory Base & Mask registers

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.





Register bit assignment

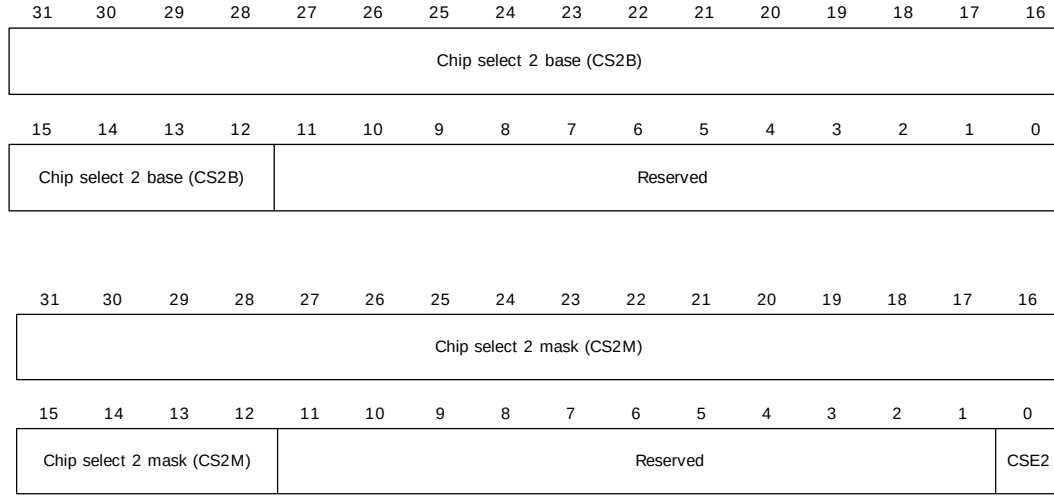
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS1B	0x50000	**Chip select 1 base** Base address for chip select 1
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS1M	0xF0000	**Chip select 1 mask** Mask or size for the chip select 1.
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE1	0x1	**Chip select 1 enable** 0 Disable chip select 1 Enable chip select

Table 77: 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.



Register bit assignment

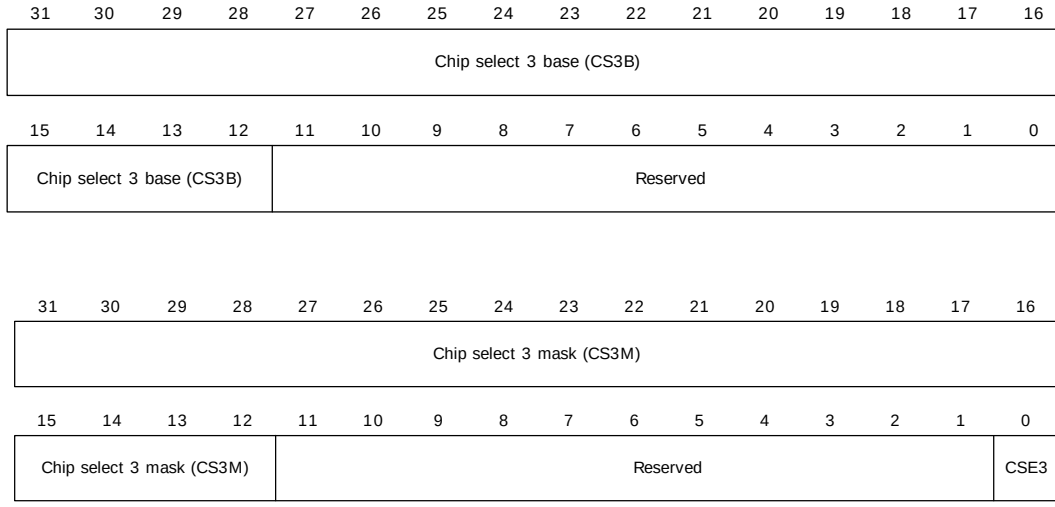
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS2B	0x60000	**Chip select 2 base** Base address for chip select 2.
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS2M	0xF0000	**Chip select 2 mask** Mask or size for chip select 2.
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE2	0x1	**Chip select 2 enable** 0 Disable chip select 1 Enable chip select

Table 78: 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

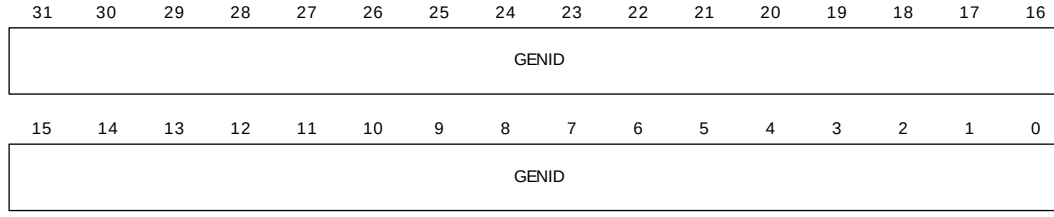
Bits	Access	Mnemonic	Reset	Description
D31:12	R/W	CS3B	0x70000	**Chip select 3 base** Base address for chip select 3.
D11:00	N/A	Reserved	N/A	N/A
D31:12	R/W	CS3M	0xF0000	**Chip select 3 mask** Mask or size for chip select 3.
D11:01	N/A	Reserved	N/A	N/A
D00	R/W	CSE3	0x1	**Chip select 3 enable** 0 Disable chip select 1 Enable chip select

Table 79: 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

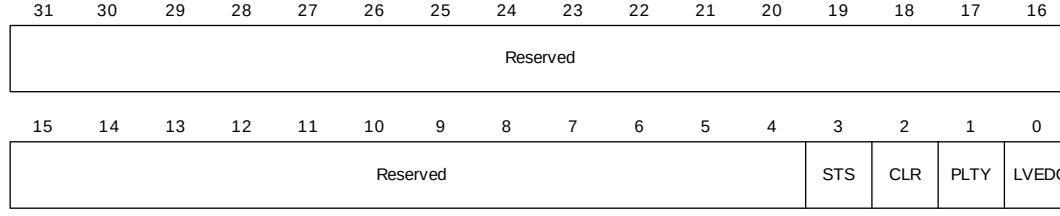
Bits	Access	Mnemonic	Reset	Description
D31:00	R	GENID	Reflects the status of the GPIO inputs at reset. The GPIO signals are listed in "Bootstrap initialization," beginning on page 153.	**GenID** General Purpose ID register

Table 80: 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 50).



Register bit assignment

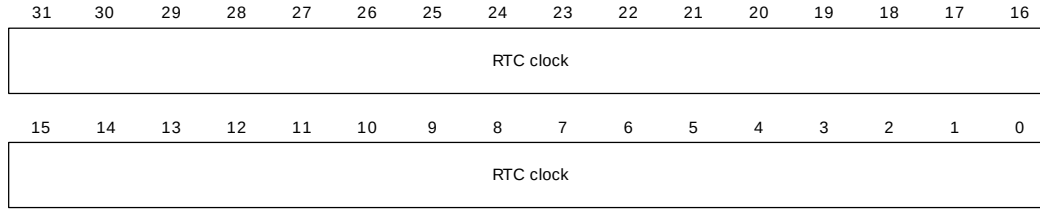
Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	STS	N/A	**Status** Status of the external signal before edge detect or level conversion.
D02	R/W	CLR	0x0	**Clear** Write a 1, then a 0 to this bit to clear the interrupt generated by the edge detect circuit.
D01	R/W	PLTY	0x0	**Polarity** 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 81: External Interrupt 0–3 Control register

RTC Clock Control register

Address: A090 0224

The RTC Clock Control register generates the 100Hz RTC clock.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	RTCCLK	0	**RTC clock** Program this value: PLL output frequency / 200 where PLL output frequency = 2xCPU clock frequency

Table 82: RTC Clock Control register

Memory Controller

CHAPTER 5

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

This figure shows the NS9360 memory controller in a sample system.

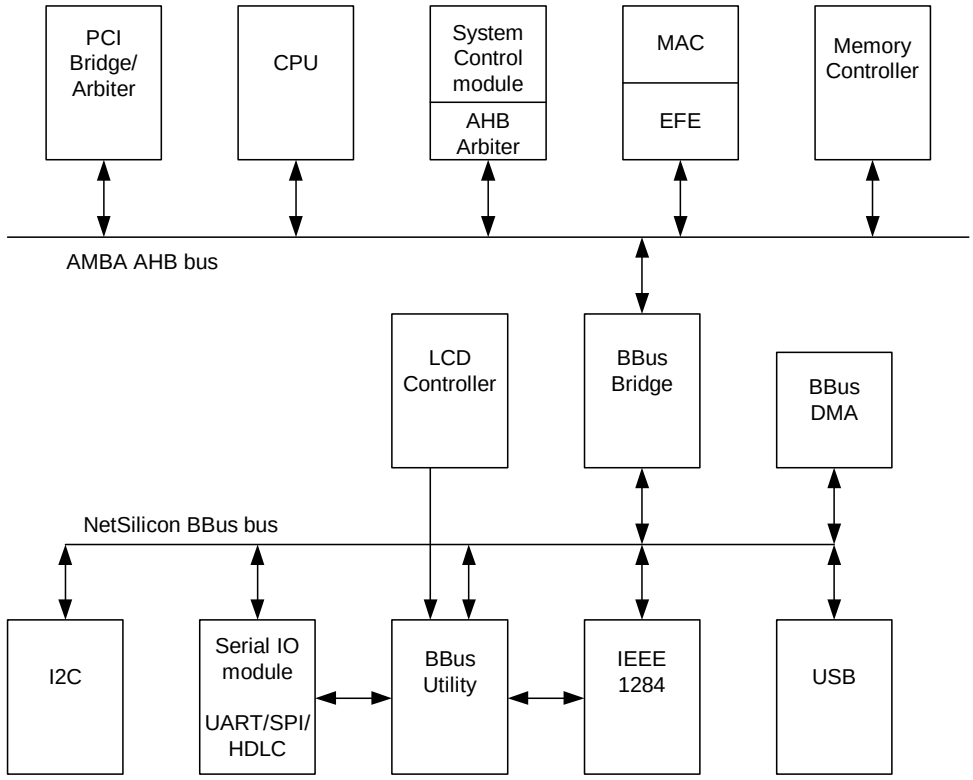


Figure 40: NS9360 sample system

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. NS9360 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.
2. Poll the SREFACK bit in the Status register.

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 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.
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.
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 83 shows configurations for the static memory controller with different types of memory devices. See "Static Memory Configuration 0–3 registers" on page 309 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 83: 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 309) 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) and Static Memory Wait Delay register (StaticWaitWr). 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 is used to time both the read and write transfers. The Static Extended Wait register allows transfers to have up to 16368 wait states.

A peripheral can, at any time, signal to the NS9360 that it wants to complete an access early by asserting the TA_STRB signal. This allows a slow peripheral with variable access times to signal that it is ready to complete an access. The NS9360 normally completes an access when it finds a rising edge on TA_STRB.

For a burst access, the peripheral must toggle TA_STRB for each access it wants to complete early. The peripheral is not required to assert TA_STRB for each access in the burst; for example, the peripheral requires the programmed access for the start of a four access burst followed by three early completion accesses, each signalled by the assertion of TA_STRB.

Using the TA_STRB signal is valid only when the EW bit is enabled.

Be aware:

- 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 309 (StaticConfig[n])
- "Static Memory Write Enable Delay 0–3 registers" on page 313 (StaticWaitWen[n])
- "Static Memory Output Enable Delay 0–3 registers" on page 314 (StaticWaitOen[n])
- "Static Memory Read Delay 0–3 registers" on page 315 (StaticWaitRd[n])
- "Static Memory Write Delay 0–3 registers" on page 317 (StaticWaitWr[n])

- "Static Memory Page Mode Read Delay 0–3 registers" on page 316 (StaticWaitPage[n])
- "Static Memory Turn Round Delay 0–3 registers" on page 318 (StaticWaitTurn[n])
- "Static Memory Extended Wait register" on page 303 (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. 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.

This figure 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 84 provides the timing parameters. Table 85 describes the transactions in Figure 41.

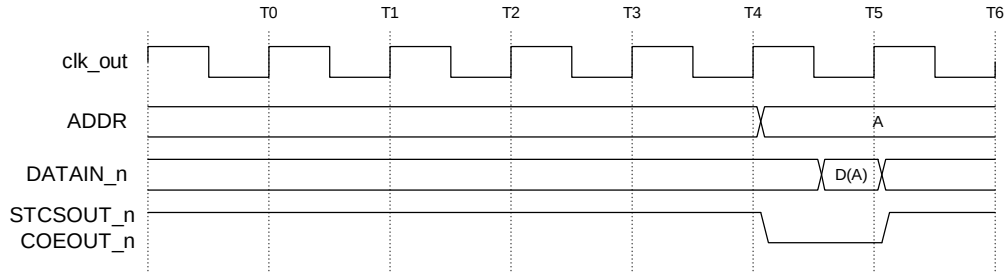


Figure 41: External memory 0 wait state read timing diagram

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

Table 84: 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 85: 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 86 provides the timing parameters. Table 87 describes the transactions in this figure.

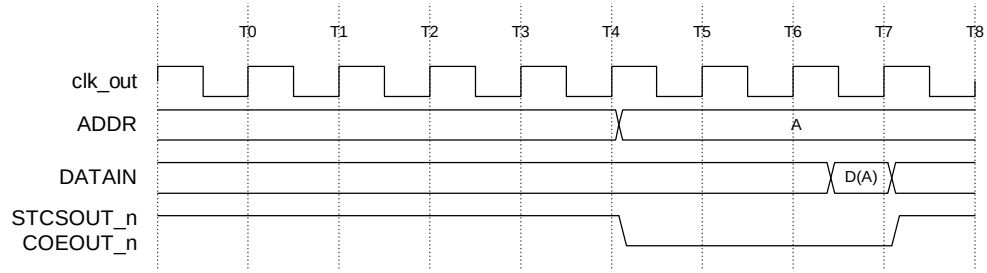


Figure 42: External memory 2 wait state read timing diagram

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

Table 86: 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.

Table 87: External memory 2 wait state read

Cycle	Description
T6-T7	Read wait state 2.
T7-T8	Read data returned from the static memory. Data is provided to the AHB.

Table 87: 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 88 provides the timing parameters. Table 89 describes the transactions for this figure.

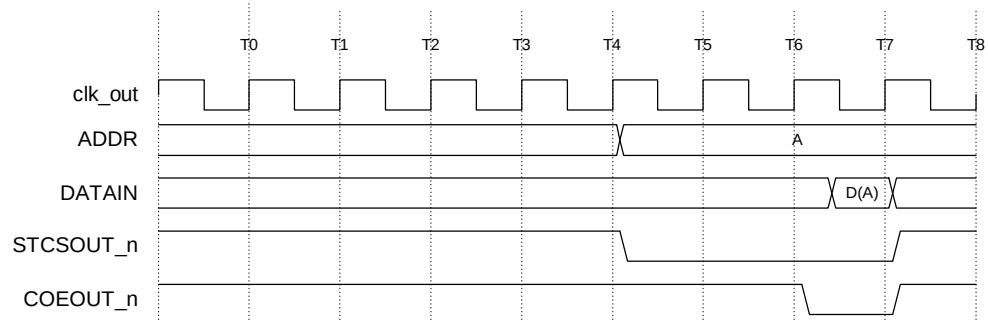


Figure 43: External memory 2 output enable delay state read timing diagram

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

Table 88: 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 89: 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 90 provides the timing parameters. Table 91 describes the transactions for this figure.

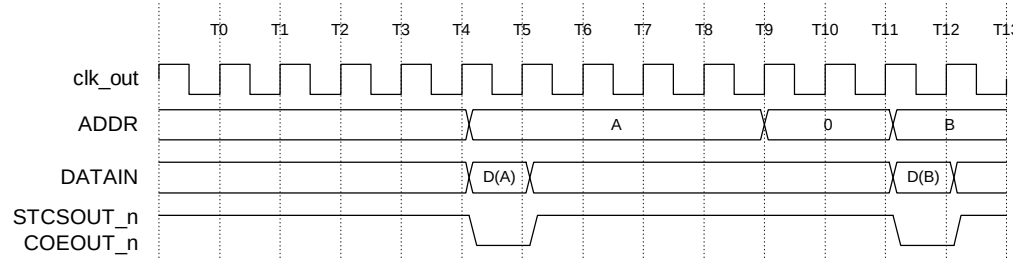


Figure 44: External memory 2 0 wait state read timing diagram

Timing parameter	Value
WAITRD	0

Table 90: Static memory timing parameters

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

Table 90: 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.
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 91: External memory 2 0 wait 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 cycles added because the external transfers are

automatically generated. Table 92 provides the timing parameters. Table 93 describes the transactions for this figure.

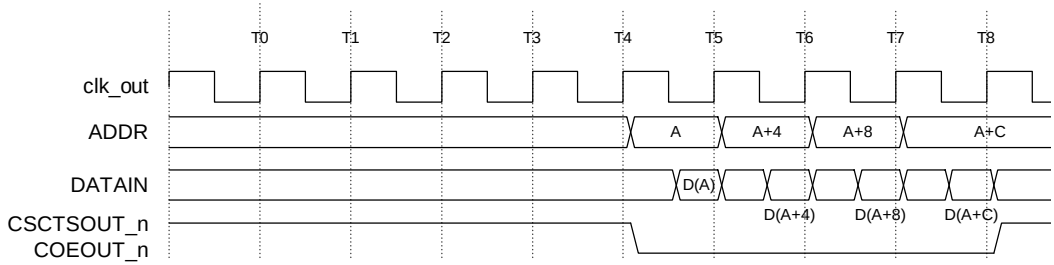


Figure 45: External memory 0 wait fixed length burst read timing diagram

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

Table 92: 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.

Table 93: External memory zero wait fixed length burst read

Cycle	Description
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 memory 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 93: 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 94 provides the timing parameters. Table 95 describes the transactions for this figure.

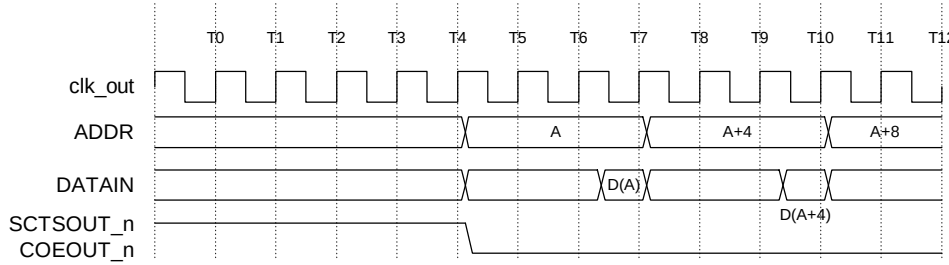


Figure 46: External memory 2 wait states fixed length burst read timing diagram

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

Table 94: 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.
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 95: 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. Table 96 provides the timing parameters. Table 97 describes the transactions for this figure.

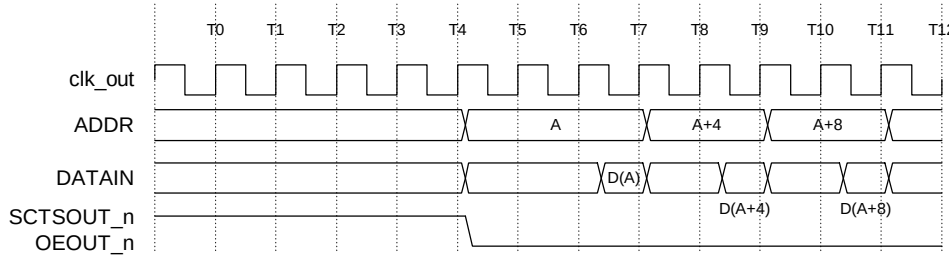


Figure 47: External memory page mode read transfer timing diagram

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

Table 96: 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.

Table 97: External memory page mode read

Cycle	Description
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.
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 97: 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 98 provides the timing parameters. Table 99 describes the transactions for this figure.

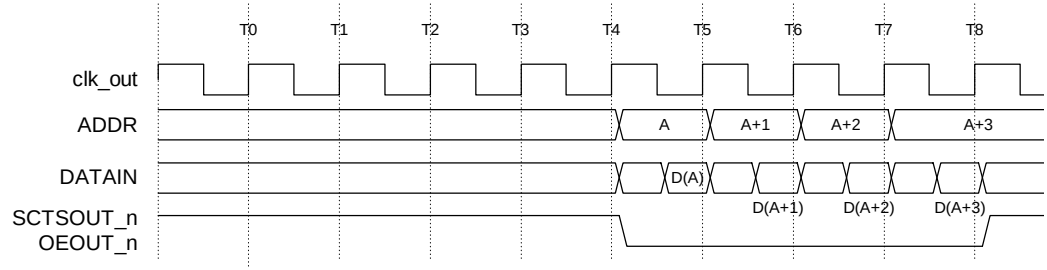


Figure 48: External memory 32-bit burst read from 8-bit memory timing diagram

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

Table 98: 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	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 99: 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 317). The WAITTURN field in the bank control register (see "Static Memory Turn Round Delay 0–3 registers" on page 318) 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 100 provides the timing parameters. Table 101 describes the transactions for this figure.

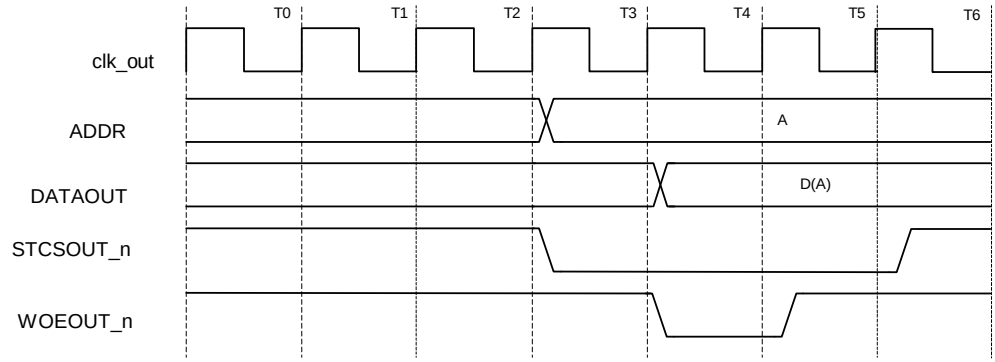


Figure 49: External memory 0 wait state write timing diagram

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

Table 100: 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 101: 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 102 provides the timing parameters. Table 103 describes the transactions for this figure.

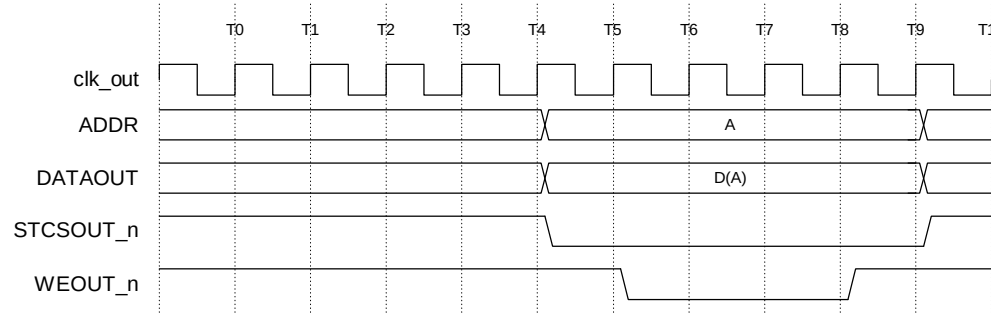


Figure 50: External memory 2 wait state write timing diagram

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

Table 102: 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 103: 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 103: 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 104 provides the timing parameters.

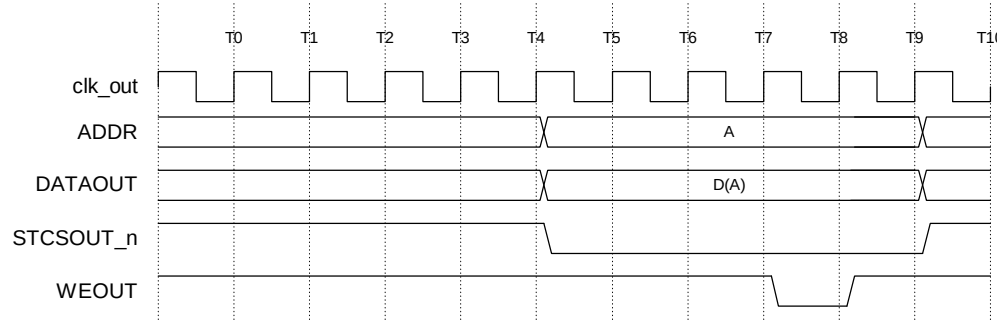


Figure 51: External memory 2 write enable delay write timing diagram

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

Table 104: 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 105: 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 106 provides the timing parameters. Table 107 describes the transactions for this figure.

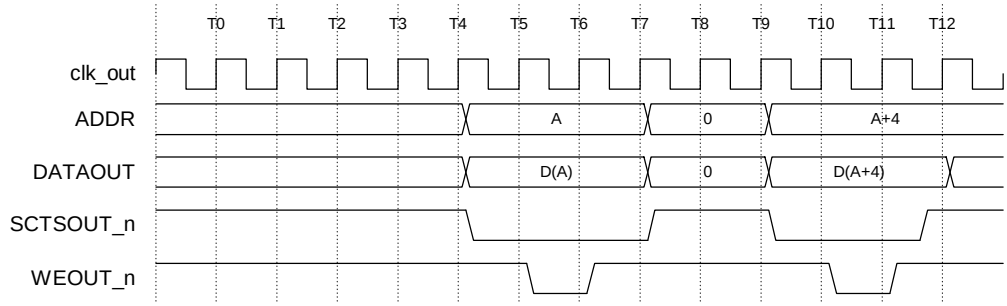


Figure 52: External memory 2 0 wait writes timing diagram

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

Table 106: 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 107: 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 107: 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. Standard AHB wait states are added to the transfers, five for the read and three for the write. The next two tables provides the timing parameters and describe the transactions for this figure.

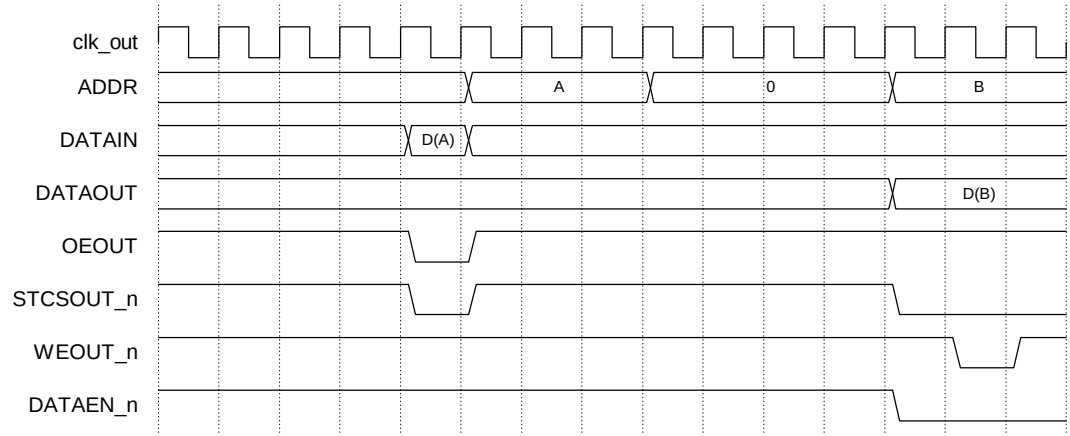


Figure 53: Read followed by write (both 0 wait) with no turnaround

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

Table 108: 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.
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 109: 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 fiveAHB 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.

The next two tables provide the timing parameters and describe the transactions for this figure.

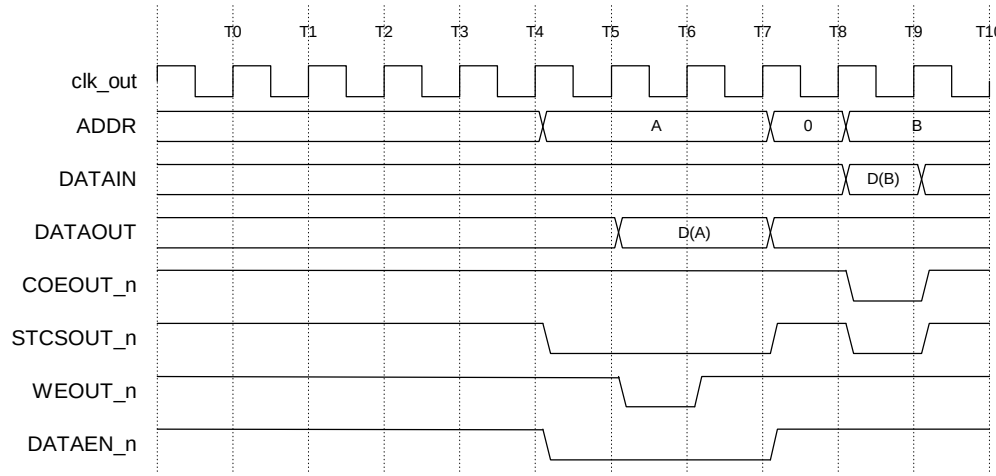


Figure 54: Write followed by a read (both 0 wait) with no turnaround

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

Table 110: 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 111: 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 111: 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 is added to the read transfer and two wait states are added to the write transfer (as for any read-write transfer sequence). The next two tables provide the timing parameters and describe the transactions for this figure.

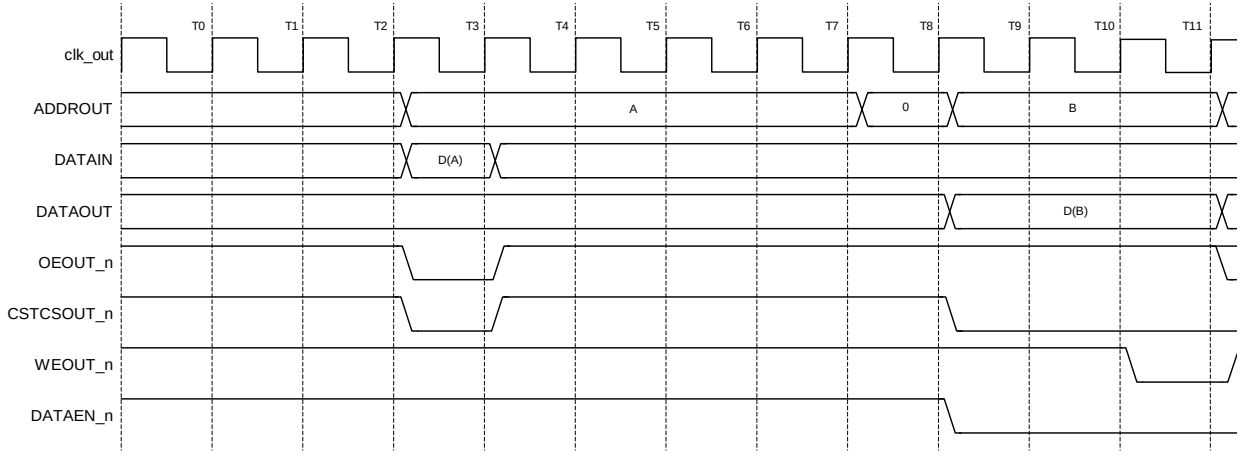


Figure 55: Read followed by a write (all 0 wait state) with two turnaround cycles

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

Table 112: 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 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 113: 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. See "Address connectivity" on page 229 for additional information, 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] is always 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.

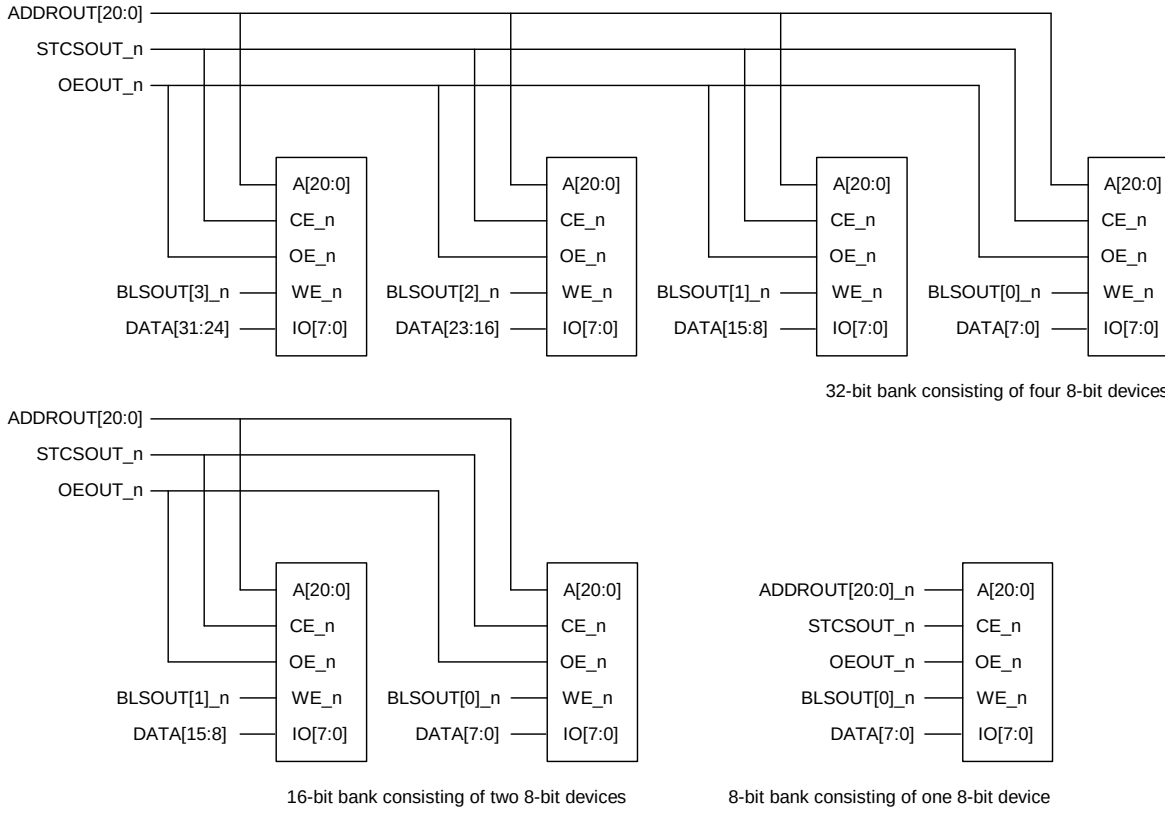


Figure 56: Memory banks constructed from 8-bit memory

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 231 and Figure 58, "Memory banks constructed from 32-bit memory," on page 231.

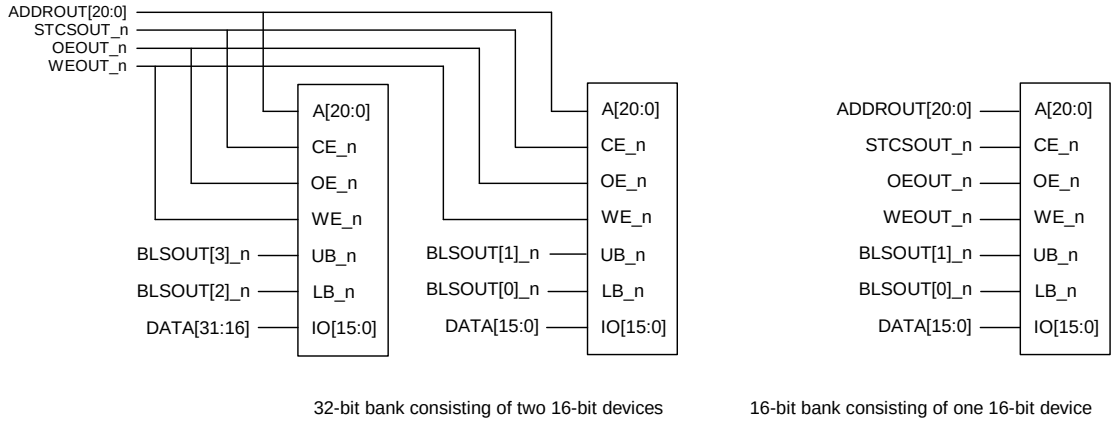


Figure 57: Memory banks constructed from 16-bit memory

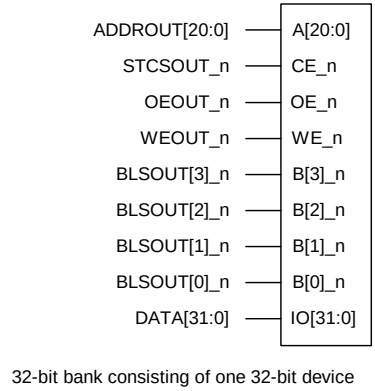


Figure 58: Memory banks constructed from 32-bit memory

The next figure shows connections for a typical memory system with different data width memory devices.

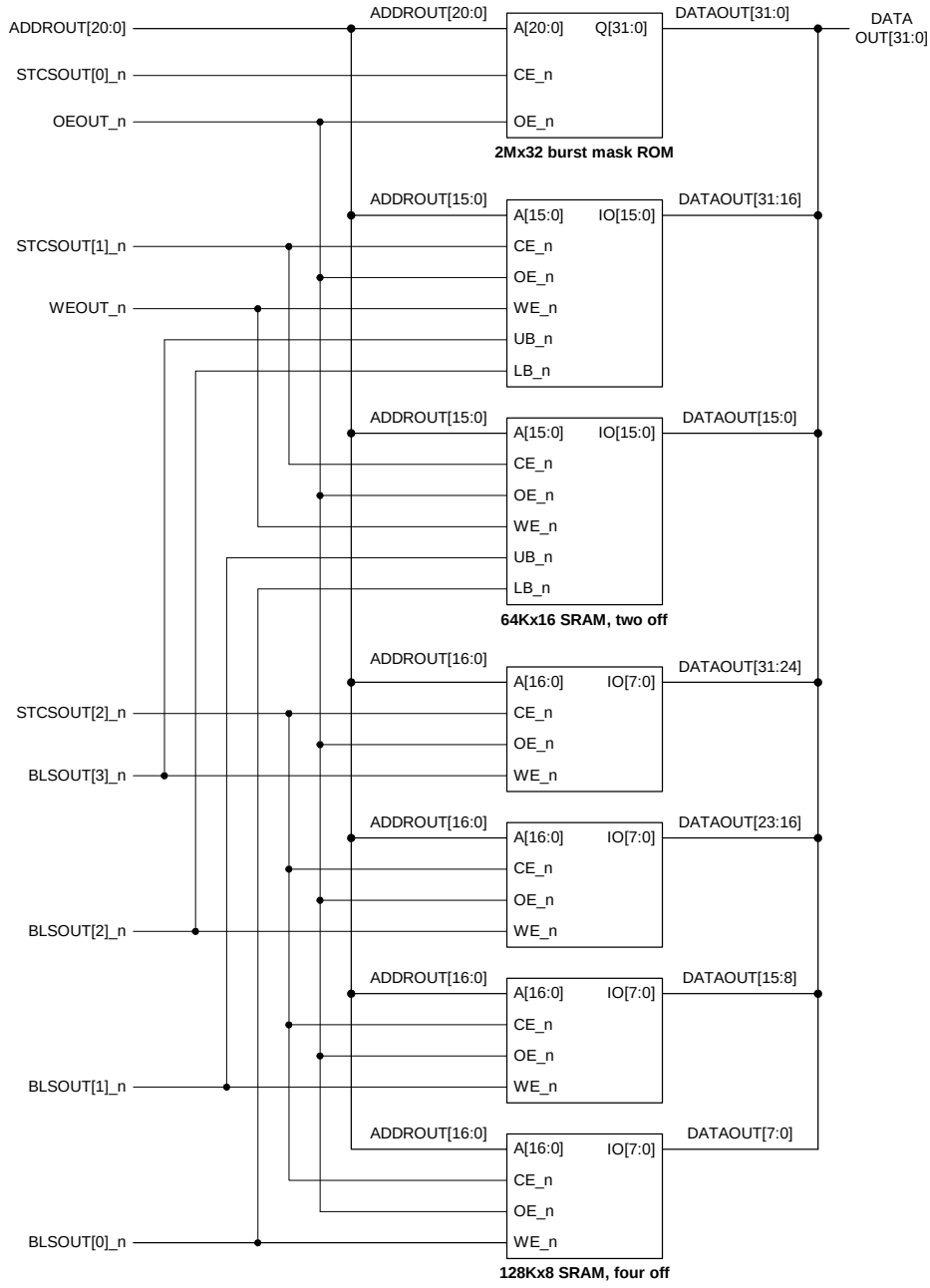


Figure 59: Typical memory connection diagram (1)

Byte lane control and databus steering

For little and big endian configurations, address right-justified

The tables in this section 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 114: 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 115: 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 116: 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 116: 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 117: 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	00	--	-	[31:24]	[23:16]
			0	00	-	-	[15:8]	[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 118: 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 119: 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 119: 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 120: 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 [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 121: 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 [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]

Table 122: 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 [1:0]	BLSOUT [3:0]	[31:24]	[23:16]	[15:8]	[7:0]
Byte	000	10	1101	-	-	[15:8]	-
Byte	000	01	1011	-	[23:16]	-	-
Byte	000	00	0111	[31:24]	-	-	-

Table 122: 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 123: 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 [0]	BLSOUT [1:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word (2 transfers	010	--	1	00	--	-	[15:8][31:24]	[7:0]
			0	00	-	-		[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 124: 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 125: 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 125: 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. 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.

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)**. 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)**. 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)**. 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)**. 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)

These tables show 32-bit wide databus address mappings for several SDRAM (RBC) devices.

Table 126 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	-
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

Table 126: Address mapping for 16M SDRAM (1Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
0	0	11	2

Table 126: Address mapping for 16M SDRAM (1Mx16, RBC)

Table 127 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
3	3	15	5
2	2	14	4
1	1	13	3
0	0	12	2

Table 127: Address mapping for 16M SDRAM (2Mx8, RBC)

Table 128 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 128: Address mapping for 64M SDRAM (2Mx32, RBC)

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	11	11
13	BA0	10	10
12	-	-	-
11	11	23	-
10	10/AP	22	AP

Table 129: Address mapping for 64M SDRAM (4Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 129: Address mapping for 64M SDRAM (4Mx16, RBC)

The next table 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

Table 130: Address mapping for 64M SDRAM (8Mx8, RBC)

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

Table 130: Address mapping for 64M SDRAM (8Mx8, RBC)

The next table 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	-	-	-
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 131: Address mapping for 128M SDRAM (4Mx32, RBC)

The next table 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
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 132: Address mapping for 128 SDRAM (8Mx16, RBC)

The next table 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	-	-

Table 133: Address mapping for 128 SDRAM (16Mx8, 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 133: Address mapping for 128 SDRAM (16Mx8, RBC)

The next table 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

Table 134: Address mapping for 256 SDRAM (8Mx32, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 134: Address mapping for 256 SDRAM (8Mx32, RBC)

The next table 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

Table 135: 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	14	3
0	0	13	2

Table 135: Address mapping for 256M SDRAM (16Mx16, RBC)

The next table 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	-
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 136: Address mapping for 256M SDRAM (32Mx8, RBC)

The next table 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
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 137: Address mapping for 512M SDRAM (32Mx16, RBC)

The next table 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

Table 138: Address mapping for 512M SDRAM (64Mx8, RBC)

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

Table 138: Address mapping for 512M SDRAM (64Mx8, RBC)

32-bit wide databus address mappings (BRC)

The next tables show 32-bit wide databus address mappings for several SDRAM (BRC) devices.

The next table 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	-

Table 139: Address mapping for 16M SDRAM (1Mx16, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 139: Address mapping for 16M SDRAM (1Mx16, BRC)

The next table 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

Table 140: Address mapping for 16M SDRAM (2Mx8, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
2	2	13	4
1	1	12	3
0	0	11	2

Table 140: Address mapping for 16M SDRAM (2Mx8, BRC)

The next table 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	-	-	-
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 141: Address mapping for 64M SDRAM (2Mx32, BRC)

The next table 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
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 142: Address mapping for 64M SDRAM (4Mx16, BRC)

The next table 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

Table 143: Address mapping for 64M SDRAM (8Mx8, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 143: Address mapping for 64M SDRAM (8Mx8, BRC)

The next table 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

Table 144: Address mapping for 128M SDRAM (4Mx32, BRC)

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

Table 144: Address mapping for 128M SDRAM (4Mx32, BRC)

The next table 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 145: Address mapping for 128M SDRAM (8Mx16, BRC)

The next table 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	-	-
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 146: Address mapping for 128M SDRAM (16Mx8, BRC)

The next table 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	-

Table 147: Address mapping for 256M SDRAM (8Mx32, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 147: Address mapping for 256M SDRAM (8Mx32, BRC)

The next table 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

Table 148: 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	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 148: Address mapping for 256M SDRAM (16Mx16, BRC)

The next table 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

Table 149: 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	13	3
0	0	12	2

Table 149: Address mapping for 256M SDRAM (32Mx8, BRC)

The next table 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 150: Address mapping for 512M SDRAM (32Mx16, BRC)

The next table 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	-
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 151: Address mapping for 512M SDRAM (64x8, BRC)

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

The following tables show 16-bit wide databus address mappings for SDRAM (RBC) devices.

The next table 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	-	-	-

Table 152: Address mapping for 16M SDRAM (1Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
12	-	-	-
11	-	-	-
10	10/AP	20	AP
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 152: Address mapping for 16M SDRAM (1Mx16, RBC)

The next table 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

Table 153: Address mapping for 16M SDRAM (2Mx8, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 153: Address mapping for 16M SDRAM (2Mx8, RBC)

The next table 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

Table 154: Address mapping for 64M SDRAM (4Mx16, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
2	2	13	3
1	1	12	2
0	0	11	**

Table 154: Address mapping for 64M SDRAM (4Mx16, RBC)

The next table 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 155: Address mapping for 64M SDRAM (8Mx8, RBC)

The next table 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	-	-	-
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 156: Address mapping for 128M SDRAM (8Mx16, RBC)

The next table 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

Table 157: Address mapping for 128M SDRAM (16Mx8, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 157: Address mapping for 128M SDRAM (16Mx8, RBC)

The next table 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

Table 158: Address mapping for 256M SDRAM (16Mx16, 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 158: Address mapping for 256M SDRAM (16Mx16, RBC)

The next table 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 159: Address mapping for 256M SDRAM (32Mx8, RBC)

The next table 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 160: Address mapping for 512M SDRAM (32Mx16, RBC)

The next table 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	-

Table 161: Address mapping for 512M SDRAM (64Mx8, RBC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 161: Address mapping for 512M SDRAM (64Mx8, RBC)

16-bit wide databus address mappings (BRC)

These tables show 16-bit wide databus address mappings for SDRAM (BRC) devices.

The next table 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	-

Table 162: Address mapping for 16M SDRAM (1Mx16, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 162: Address mapping for 16M SDRAM (1Mx16, BRC)

The next table 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
9	9	19	-
8	8	18	9
7	7	17	8
6	6	16	7
5	5	15	6
4	4	14	5

Table 163: Address mapping for 16M SDRAM (2Mx8, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
3	3	13	4
2	2	12	3
1	1	11	2
0	0	10	**

Table 163: Address mapping for 16M SDRAM (2Mx8, BRC)

The next table 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
4	4	13	5
3	3	12	4
2	2	11	3
1	1	10	2
0	0	9	**

Table 164: Address mapping for 64M SDRAM (4Mx16, BRC)

The next table 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 165: Address mapping for 64M SDRAM (8Mx8, BRC)

The next table 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	-	-	-

Table 166: Address mapping for 128M SDRAM (8Mx16, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 166: Address mapping for 128M SDRAM (8Mx16, BRC)

The next table 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	-	-	-
11	11	22	-
10	10/AP	21	AP
9	9	20	10
8	8	19	9
7	7	18	8

Table 167: Address mapping for 128M SDRAM (16Mx8, BRC)

Output address (ADDROUT)	Memory device connections	AHB address to row address	AHB address to column address
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 167: Address mapping for 128M SDRAM (16Mx8, BRC)

The next table 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
6	6	16	7
5	5	15	6
4	4	14	5
3	3	13	4
2	2	12	3

Table 168: 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	11	2
0	0	10	**

Table 168: Address mapping for 256M SDRAM (16Mx16, BRC)

The next table 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
1	1	12	2
0	0	11	**

Table 169: Address mapping for 256M SDRAM (32Mx8, BRC)

The next table 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 170: Address mapping for 512M SDRAM (32Mx16, BRC)

The next table 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

Address mapping for 512M SDRAM (64Mx8, BRC)

Output address (ADDROUT**)**	**Memory device connections**	**AHB address to row address**	**AHB address to column address**
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	**

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 171 lists the registers in the Memory Controller module.

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 171: 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 171: 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 171: Memory Controller register map

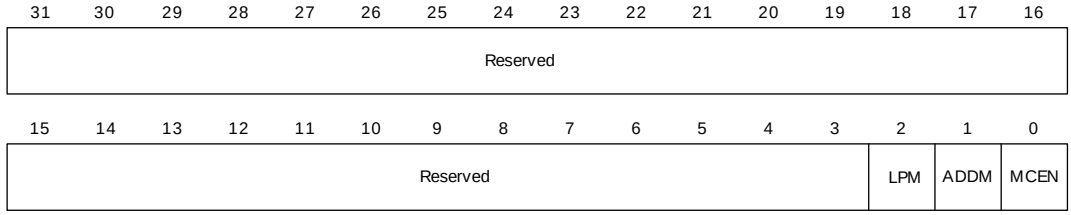
Reset values

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.



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 172: Control register

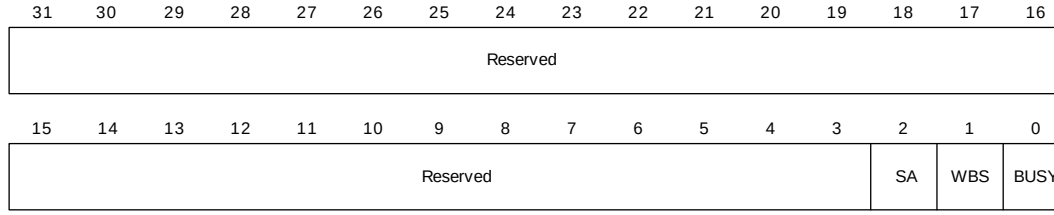
Bits	Access	Mnemonic	Description
D01	R/W	ADDM	**Address mirror** 0 Normal memory map 1 Reset memory map. Static memory chip select 1 is mirrored onto chip select 0 and chip select 4 (reset value on reset_n) Indicates normal or reset memory map. On power-on reset, chip select 1 is mirrored to both chip 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 172: Control register

Status register

Address: A070 0004

The Status register provides memory controller status information.



Register bit assignment

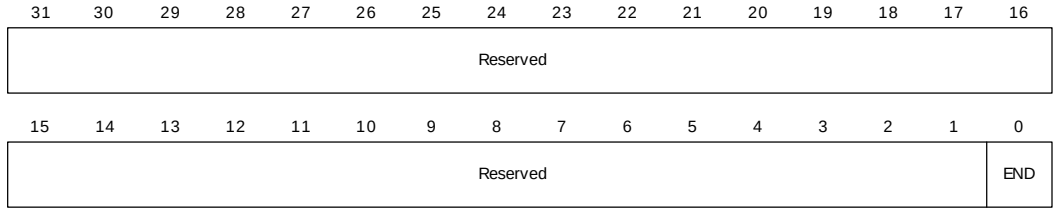
Bits	Access	Mnemonic	Description
D31:03	N/A	Reserved	N/A (do not modify)
D02	R	SA	**Self-refresh acknowledge (SREFACK)** 0 Normal mode 1 Self refresh mode (reset value on reset_n) Indicates the memory controller operating mode.
D01	R	WBS	**Write buffer status** 0 Write buffers empty (reset value on reset_n) 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 173: 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.



Register bit assignment

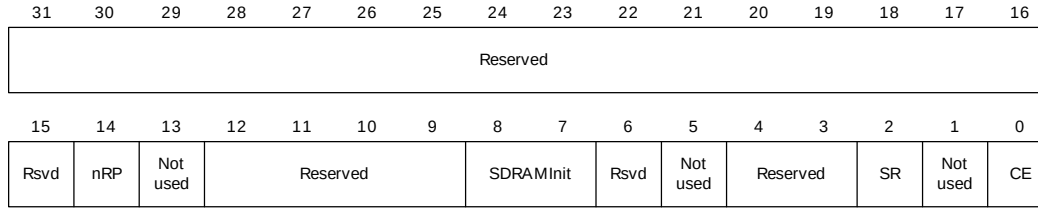
Bits	Access	Mnemonic	Description
D31:01	N/A	Reserved	N/A (do not modify)
D00	R/W	END	Endian mode 0 Little endian mode 1 Big endian mode The value of the endian bit on power-on reset (reset_n) is determined by the gpio[44] signal. This value can be overridden by software. This field is not affected by 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 174: 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.



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	Always write to 0.
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 175: 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 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.

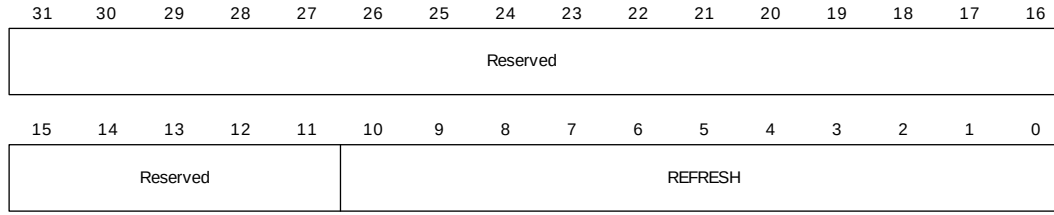
Table 175: Dynamic Memory Control register

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.



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 176: 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:

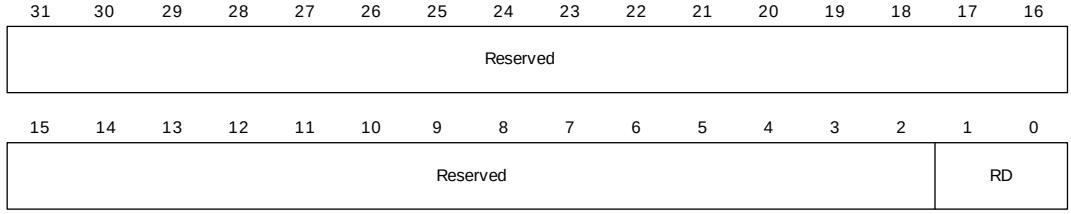
- The refresh cycles are evenly distributed. There might be slight variations, however, when the auto-refresh command is issued, depending on the status of the memory controller.
- Unlike other SDRAM memory timing parameters, the refresh period is programmed in the CPU_CLK domain.

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).

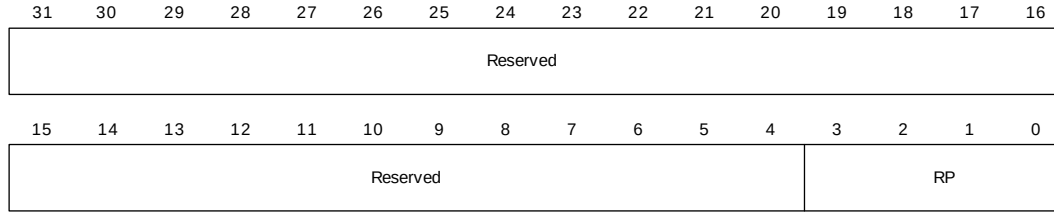
Table 177: Dynamic Memory Read Configuration register

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)

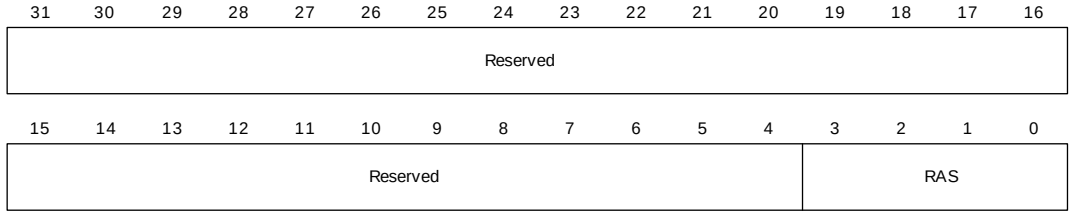
Table 178: Dynamic Memory Precharge Command Period register

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)

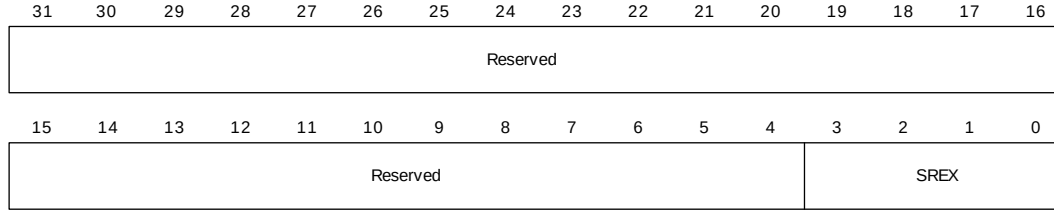
Table 179: Dynamic Memory Active to Precharge Command Period register

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.



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)

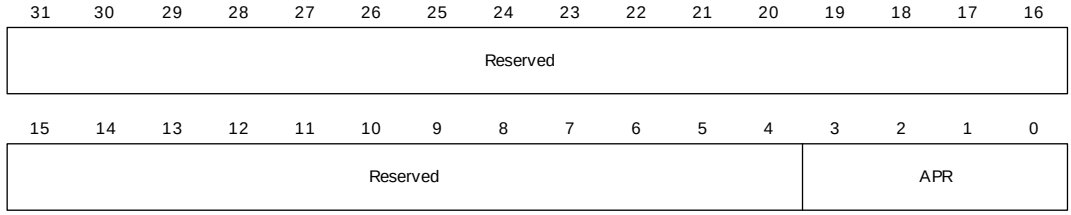
Table 180: Dynamic Memory Self-refresh Exit Time register

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)

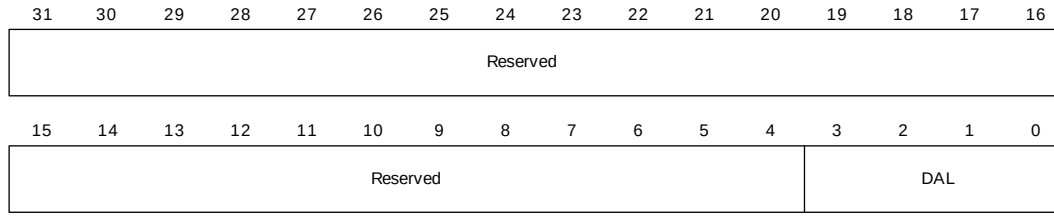
Table 181: Dynamic Memory Last Data Out to Active Time register

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** 16 clock cycles (reset value on reset_n)

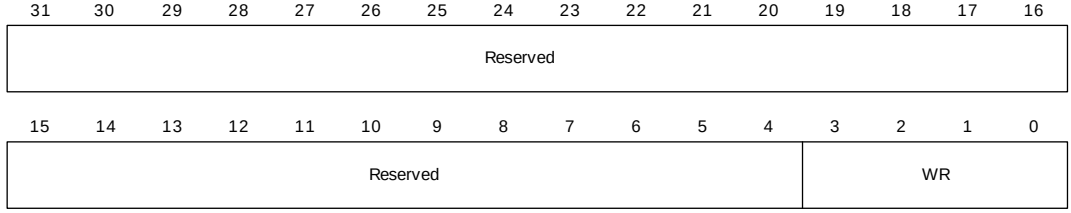
Table 182: Dynamic Memory Data-in Active Command Time register

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.



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)

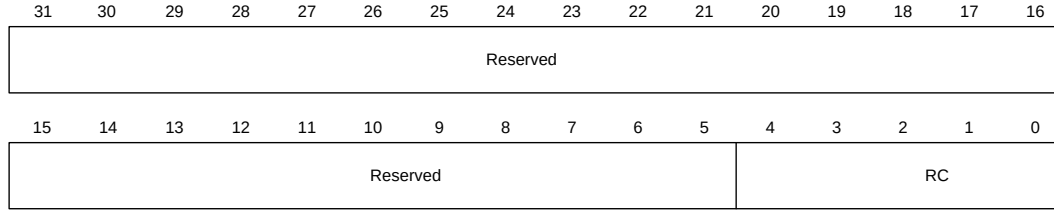
Table 183: Dynamic Memory Write Recovery TIme register

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)

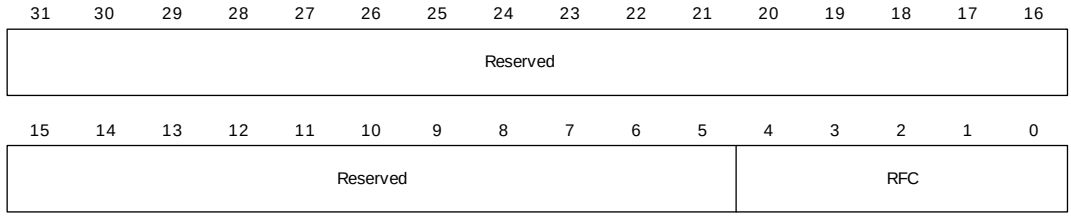
Table 184: Dynamic Memory Active to Active Command Period register

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.



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)

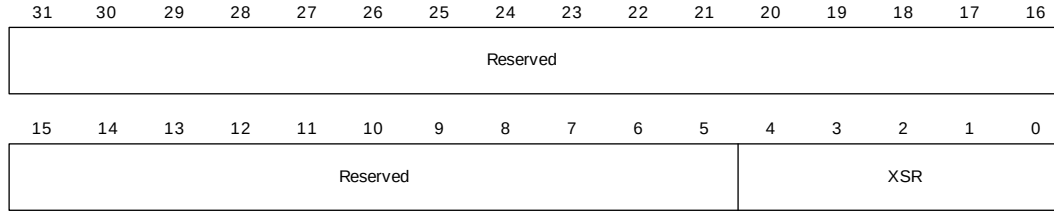
Table 185: Dynamic Memory Auto Refresh Period register

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)

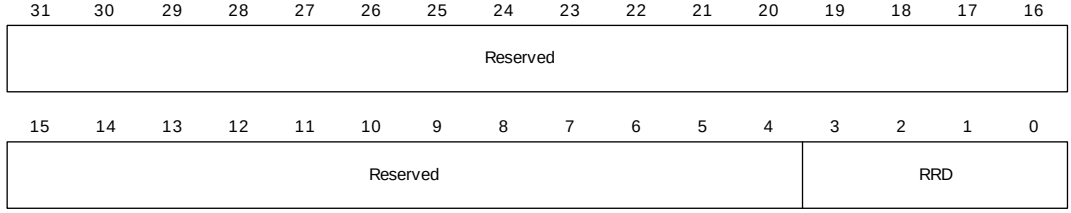
Table 186: Dynamic Memory Exit Self-refresh register

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)

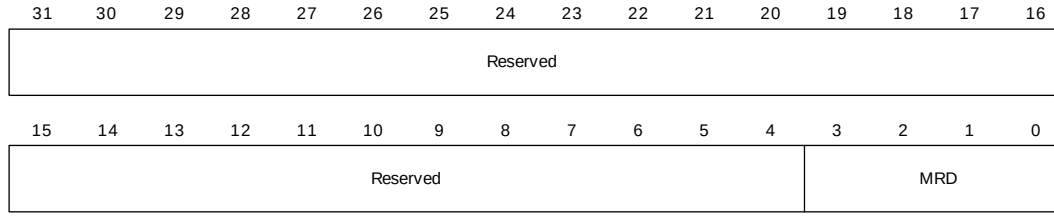
Table 187: Dynamic Memory Active Bank A to Active Bank B Time register

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 188: Dynamic Memory Load Mode register to Active Command Time register

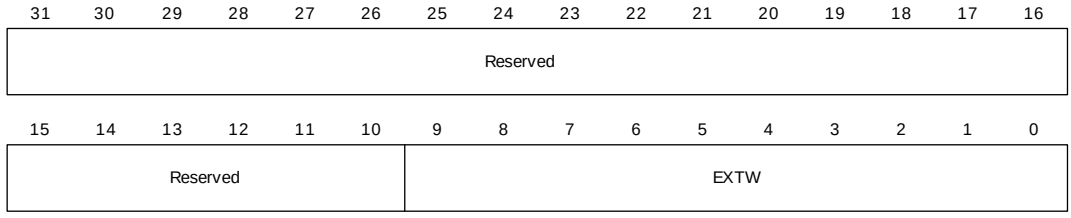
Static Memory Extended Wait register

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 or the Static Memory Write Delay registers) when the EW (extended wait) bit in the related Static Memory Configuration register 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 189: Static Memory Extended Wait register

Example

Static memory read/write time = 16 µs

CLK frequency = 50 MHz

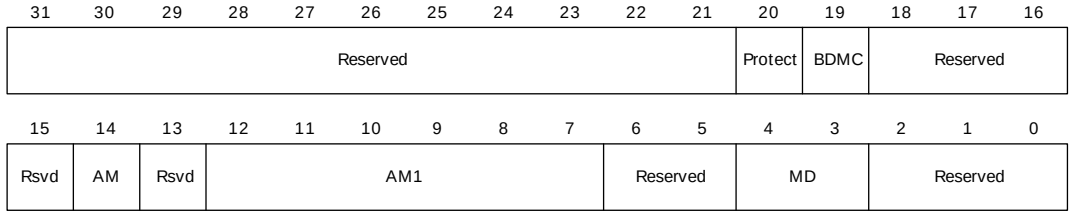
This value must be programmed into the Static Memory Extended Wait register:

$(16 \times 10^{-6} \times 50 \times 10^{6} / 16) - 1 = 49$

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.



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 190: Dynamic Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D14	R/W	AM	**Address mapping** 0 Reset value on reset_n See Table 191, “Address mapping,” on page 305 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 191, “Address mapping,” on page 305 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 190: Dynamic Memory Configuration 0–3 registers

The next table 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 191: 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 191: 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 191: 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

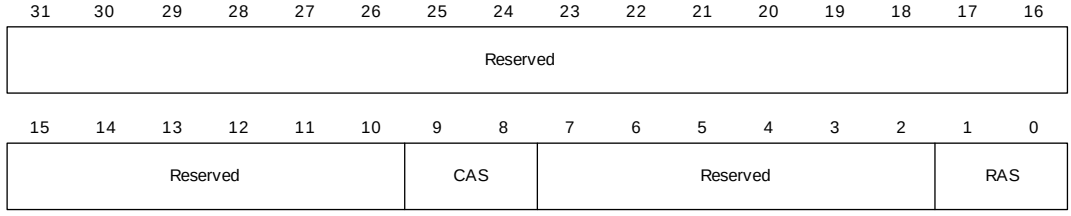
For a chip select connected to	Select this mapping
32-bit wide memory device	32-bit wide address mapping
16-bit wide memory device	16-bit wide address mapping
4 x 8-bit wide memory devices	32-bit wide address mapping
2 x 8-bit memory devices	16-bit wide address mapping

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.



Register bit assignment

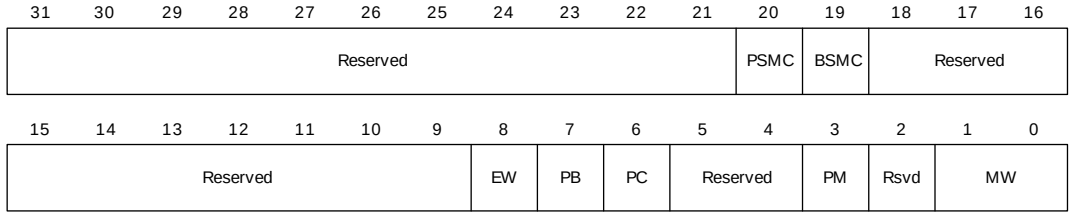
Bits	Access	Mnemonic	Description
D31:10	N/A	Reserved	N/A (do not modify)
D09:08	R/W	CAS	**CAS latency** 00 Reserved 01 One clock cycle, where the RAS to CAS latency (RAS) and CAS latency (CAS) are defined in CLK cycles 10 Two clock cycles 11 Three clock cycles (reset value on reset_n)
D07:02	N/A	Reserved	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 192: 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.



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 193: Static Memory Configuration 0–3 registers

Bits	Access	Mnemonic	Description
D08	R/W	EW	**Extended wait** 1 Extended wait disabled (reset value on reset_n) 1 Extended wait enabled Extended wait uses the Static Extended Wait register to time both the read and write transfers, rather than the Static Memory Read Delay 0–3 registers and Static Memory Write Delay 0–3 registers. 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 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 193: 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 193: 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.

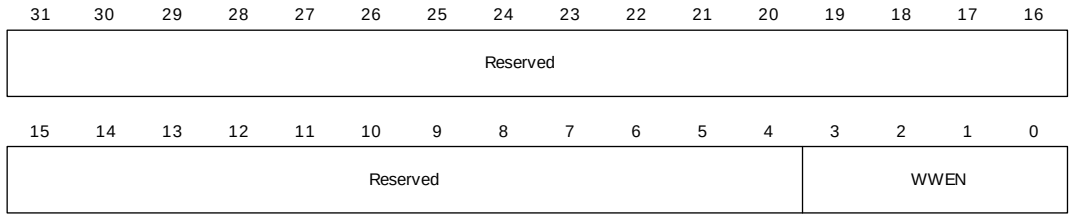
Table 193: Static Memory Configuration 0–3 registers

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.



Register bit assignment

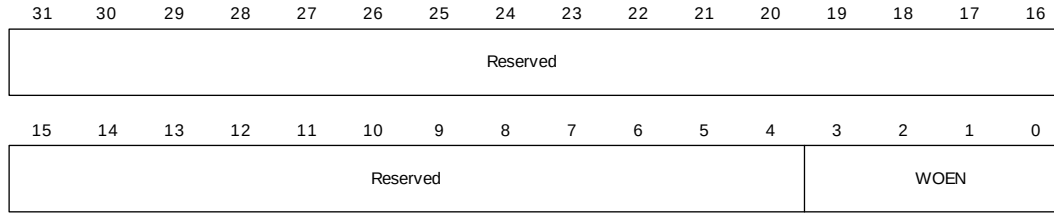
Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WWEN	**Wait write enable (WAITWEN)** 0000 One HCLK cycle delay between assertion of chip select and write enable (reset value on reset_n). 0001–1111 (n+1) HCLK cycle delay, where the delay is (WAITWEN+1) x t_{HCLK} Delay from chip select assertion to write enable.

Table 194: 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

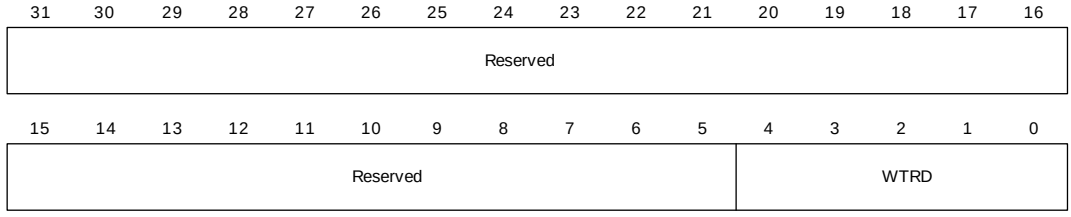
Bits	Access	Mnemonic	Description
D31:04	N/A	Reserved	N/A (do not modify)
D03:00	R/W	WOEN	**Wait output enable (WAITOEN)** 0000 No delay (reset value on reset_n). 0001–1111 n cycle delay, where the delay is WAITOEN x t_{HCLK} Delay from chip select assertion to output enable.

Table 195: 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.



Register bit assignment

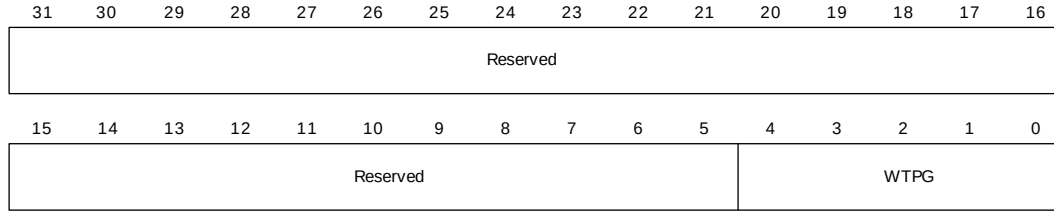
Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTRD	**Nonpage mode read wait states or asynchronous page mode read first access wait state (WAITRD)** 00000–11110 (n+1) HCLK cycle for read accesses. For nonsequential reads, the wait state time is (WAITRD+1) x t_{HCLK} 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 196: 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.



Register bit assignment

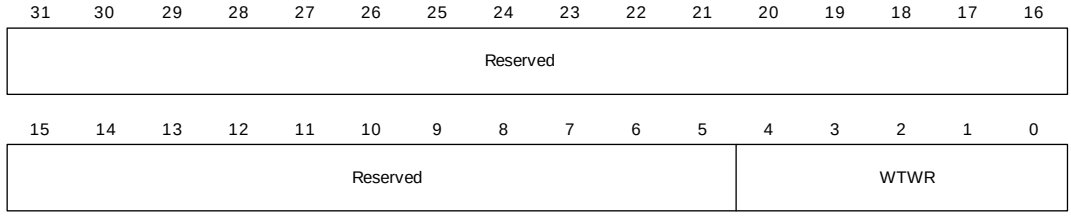
Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTPG	**Asynchronous page mode read after the first wait state (WAITPAGE)** 00000–11110 (n+1) HCLK cycle for read access time. For asynchronous page mode read for sequential reads, the wait state time for page mode accesses 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 197: 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.



Register bit assignment

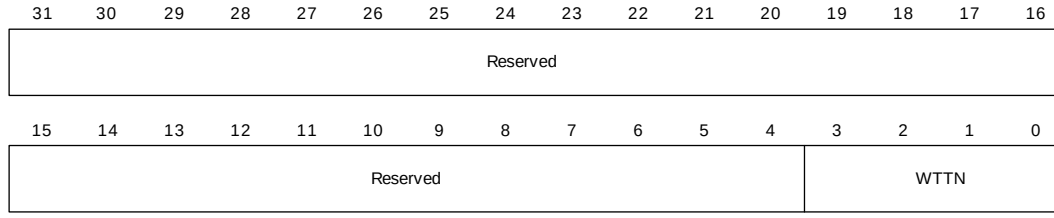
Bits	Access	Mnemonic	Description
D31:05	N/A	Reserved	N/A (do not modify)
D04:00	R/W	WTWR	**Write wait states (WAITWR)** 00000–11110 (n+2) HCLK cycle write access time. The wait state time for write accesses after the first read is WAITWR (n+2) x t_{HCLK} 11111 332 HCLK cycle write access time (reset value on reset_n) SRAM wait state time for write accesses after the first read.

Table 198: 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.



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 199: 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.

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: Media Independent Interface (MII) and Reduced Media Independent Interface (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 60 shows the Ethernet Communications module.

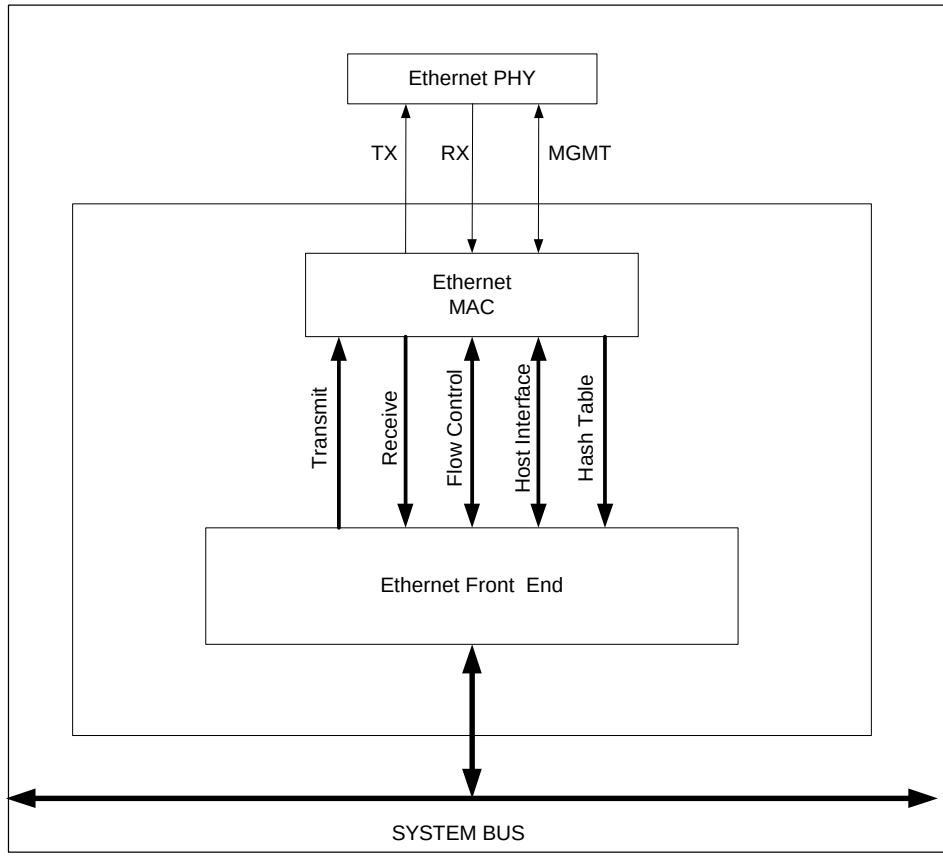


Figure 60: Ethernet Communication module block diagram

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 61 shows the Ethernet MAC module block diagram, with its associated hierarchy. Table 200 describes the module's features.

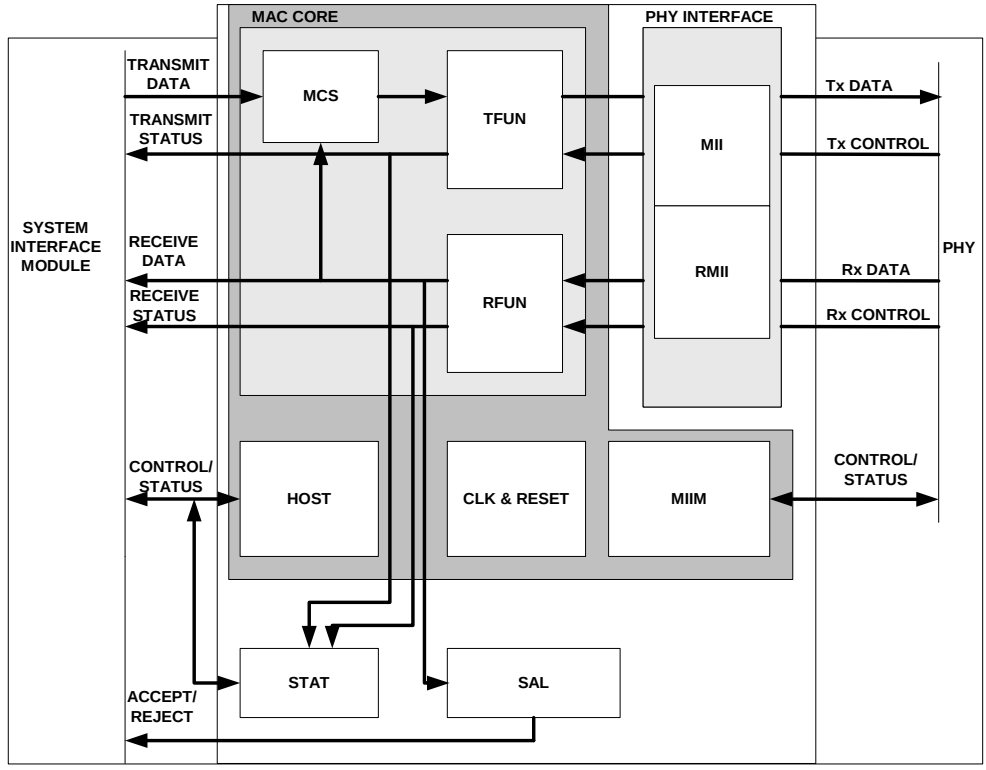


Figure 61: Ethernet MAC block diagram

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.
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.

Table 200: Ethernet MAC features

Table 201 shows how the different PHY interfaces are mapped to the external IO. In addition to these signals, NS9360 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 NS9360
RXD[2]	RXD[2]	N/C Pull low external to NS9360

Table 201: PHY interface mappings to external IO

External IO	MII	RMII
RXD[1]	RXD[1]	RXD[1]
RXD[0]	RXD[0]	RXD[0]
RX_DV	RX_DV	N/C Pull low external to NS9360
RX_ER	RX_ER	RX_ER Optional signal; pull low external to NS9360 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 NS9360
CRS	CRS	CRS_DV
COL	COL	N/C Pull low external to NS9360
MDC	MDC	MDC
MDIO	MDIO	MDIO

Table 201: 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 371).

- Accept frames to destination address programmed in the SA1, SA2, and SA3 registers (Station Address registers, beginning on page 369)
- Accept all frames
- Accept all multicast frames
- Accept all multicast frames using HT1 and HT2 registers (Hash Table registers, beginning on page 372)
- 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 372). 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 404, 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 374).

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 383). Any statistics counter overflow can cause the STOVFL bit in the Ethernet Interrupt Status register (see page 391) to be set if its associated mask bit is not set in Carry Mask Register 1 or Carry Mask Register 2.

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 62 shows the Ethernet front-end module (EFE).

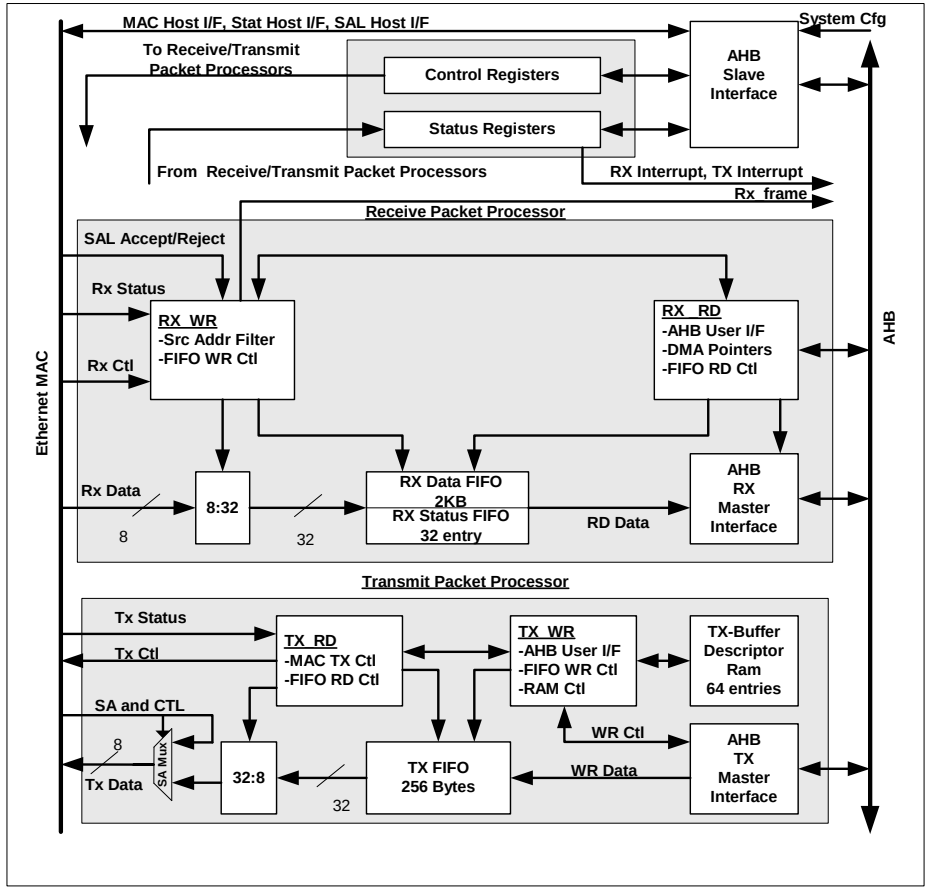


Figure 62: Ethernet front-end module block diagram

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 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 to 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 NS9360'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 63 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 63: 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 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 ENABLE 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. **Note:** To enable a pool that is currently disabled, change the ENABLE bit from 0 to 1 and reinitialize the buffer descriptors pointed to by the Buffer Descriptor Pointer register: 1 Set the ERXINIT bit in the Ethernet General Control Register 1. 2 Wait for RXINIT to be set in the Ethernet General Status register. Change the ENABLE bit only while the receive packet processor is idle.
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.

Field	Description
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.
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 64 shows the transmit buffer descriptor format.

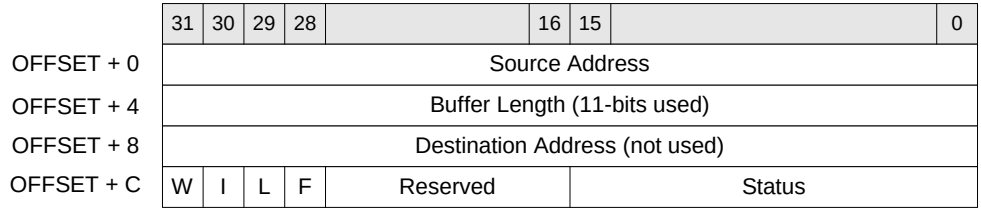


Figure 64: Transmit buffer descriptor format

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). When the WRAP bit is not set, the next buffer descriptor is located at the next entry in the TX buffer descriptor RAM.
I	When set, tells the TX_WR logic to set TXBUFC in the Ethernet Interrupt Status register 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.

Field	Description
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. 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.

- 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 397). 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 TX 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 348) and written into the status field of the current buffer descriptor.

Note:

An Ethernet underrun can only occur due to the following programming errors:

- Insufficient bandwidth is assigned to the Ethernet transmitter.
- A packet consisting of multiple, linked buffer descriptors does not have the F bit set in any of the non-first buffer descriptors.

When an underrun occurs, it is also possible for the Ethernet transmitter to send out a corrupted packet with a good Ethernet CRC if the MAC is configured to add the CRC to the frame (that is, CRCEN in MAC Configuration Register #2 is set to 1).

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 179).

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. 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 202 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

Table 202: Ethernet interrupt conditions

Interrupt condition	Description	Interrupt
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
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 202: 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 203 provides a summary of all resets used for the Ethernet front-end and MAC, as well as the modules the resets control. See the NS9360 *Sample Driver Configurations* for examples of how the resets are used.

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
RMIIM	MII Management Configuration register	1	0	MAC MIIM logic
RPERMII	PHY Support register	1	0	RMII

Table 203: Reset control

External CAM filtering

NS9360 supports external Ethernet CAM filtering, which requires an external CAM controller to operate in conjunction with the MAC inside NS9360. The interface to the CAM controller is provided through GPIO in NS9360.

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 NS9360, to identify the beginning of each Ethernet frame being transferred to NS9360. 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 65 shows the timing relationship between the CAM_REQ, CAM_REJECT, and MII receive interface signals when using an MII PHY.

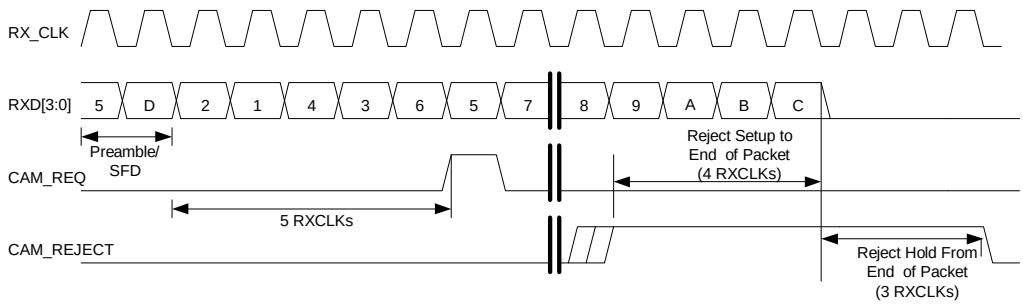
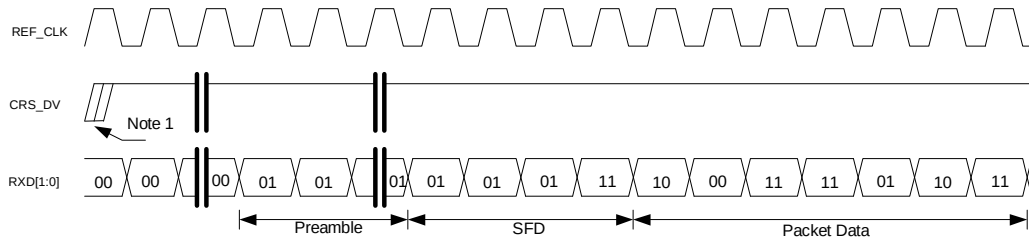


Figure 65: External Ethernet CAM filtering for MII PHY

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.

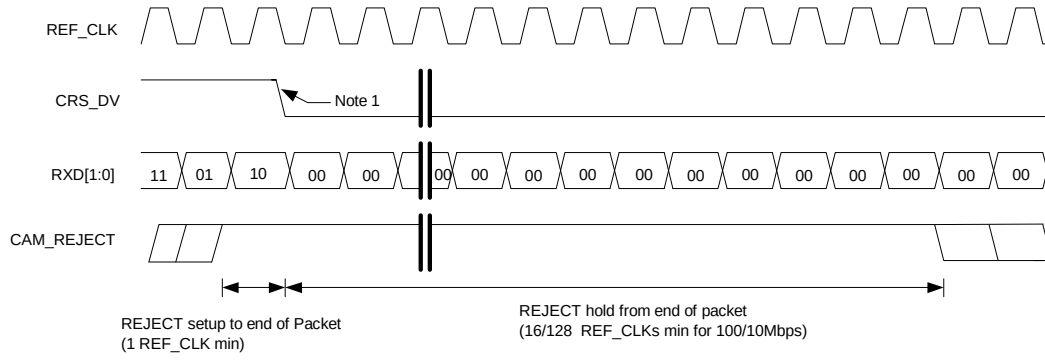
For RMII PHYs, the external CAM filtering logic is different, because the PHY interface is 2 bits at 50 MHz rather than the 4 bits at 25 MHz for a MII PHY. Because the CAM_REQ signal is generated from the 25 MHz clock, it cannot be used reliably with external 50 MHz logic to identify the start of a new 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 66. 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.



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 66: 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 67. 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).



1. Falling edge of CRS_DV synchronous relative to REF_CLK.

Figure 67: External Ethernet CAM filtering for RMII PHY

Ethernet Control and Status registers

Table 204 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

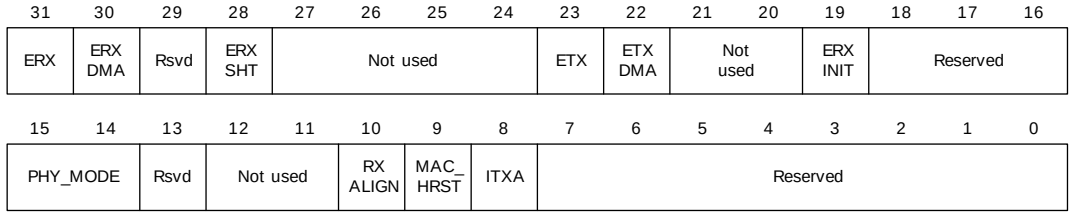
Table 204: Ethernet Control and Status register map

Address	Register	Description
A060 0444	SA2	Station Address Register #2
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 204: Ethernet Control and Status register map

Ethernet General Control Register #1

Address: A060 0000



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	ERX	0	**Enable RX packet processing** 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 205: 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** 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, 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 205: 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 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 NS9360. NS9360 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.

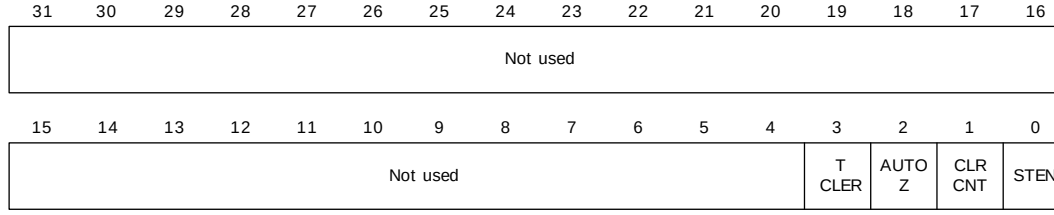
Table 205: Ethernet General Control Register #1

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 205: Ethernet General Control Register #1

Ethernet General Control Register #2

Address: A060 0004



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	Not used	0	Always write as 0.

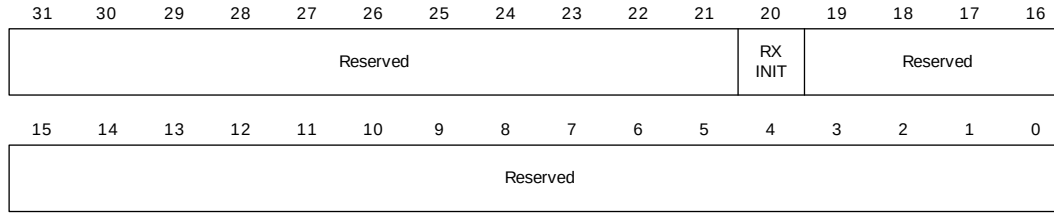
Table 206: Ethernet General Control Register #2

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). 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 206: Ethernet General Control Register #2

Ethernet General Status register

Address: A060 0008



Register bit assignment

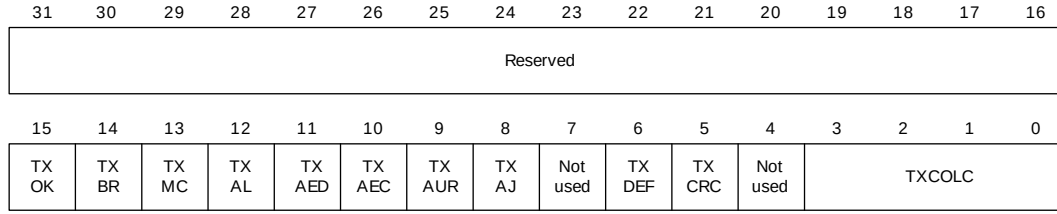
Bits	Access	Mnemonic	Reset	Description
D31:21	N/A	Reserved	N/A	N/A
D20	R/C	RXINIT	0x0	**RX initialization complete** Set when the RX_RD logic has completed the initialization of the local buffer descriptor registers requested when ERXINIT in Ethernet General Control Register #1 is set. The delay from ERXINIT set to RXINIT set is less than five microseconds.
D19:00	N/A	Reserved	N/A	N/A

Table 207: 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 391) 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.



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. 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. If TXAED is set, the TX_WR logic stops processing frames and sets the TXERR bit in the Ethernet Interrupt Status register.

Table 208: 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. TXAJ is set only if the HUGE bit in MAC Configuration Register #2 is set to 0. Jumbo frames result in the TX buffer descriptor buffer length field 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.
D04	R	Not used	0x0	Always set to 0.

Table 208: Ethernet Transmit Status register

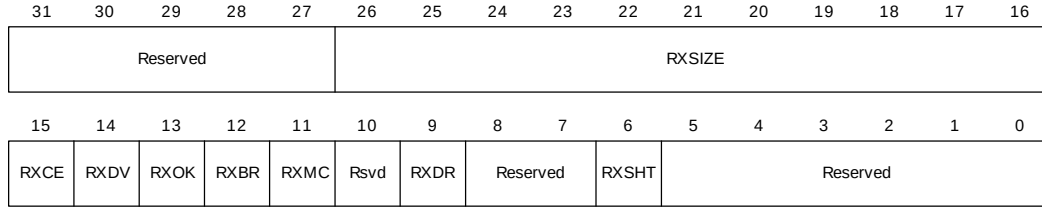
Bits	Access	Mnemonic	Reset	Description
D03:00	R	TXCOLC	0x0	**Transmit collision count** Number of collisions the frame incurred during transmission attempts.

Table 208: 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 391) 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.



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.
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.

Table 209: Ethernet Receive Status register

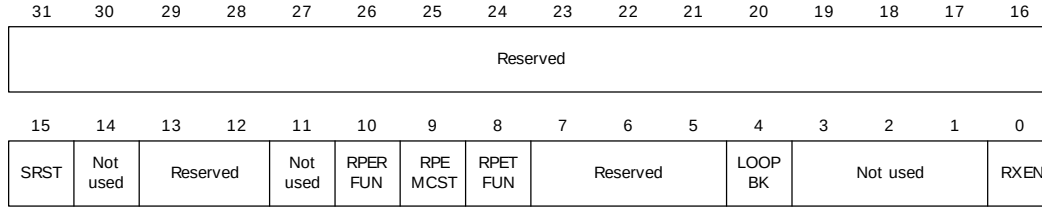
Bits	Access	Mnemonic	Reset	Description
D13	R	RXOK	0x0	**Receive frame OK** Set when the frame has a valid CRC and no symbol errors.
D12	R	RXBR	0x0	**Receive broadcast frame** Set when the frame has a valid broadcast address.
D11	R	RXMC	0x0	**Receive multicast frame** Set when the frame has a valid multicast 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.
D08:07	N/A	Reserved	N/A	N/A
D06	R	RXSHT	0x0	**Receive frame is too short** Set when the frame's length is less than 64 bytes. Short frames are accepted only when the ERXSHT bit is set to 1 in Ethernet General Control Register #1.
D05:00	N/A	Reserved	N/A	N/A

Table 209: 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.



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	R/W	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

Table 210: MAC Configuration Register #1

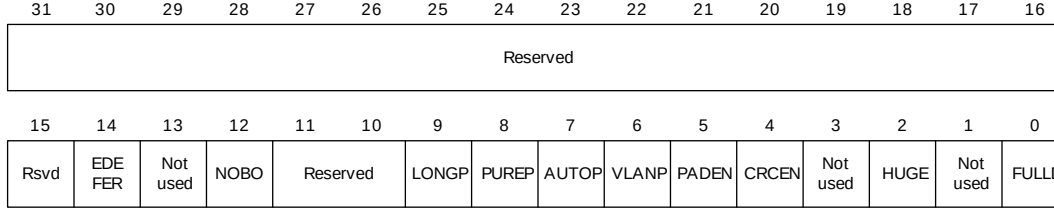
Bits	Access	Mnemonic	Reset	Description
D04	R/W	LOOPBK	0	**Internal loopback** Set this bit to 1 to cause the MAC transmit interface to be internally looped back to the MAC receive interface. Clearing this bit results in normal operation.
D03:01	R/W	Not used	0	Always write as 0.
D00	R/W	RXEN	0	**Receive enable** Set this bit to 1 to allow the MAC receiver to receive frames.

Table 210: 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.



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.

Table 211: MAC Configuration Register #2

Bits	Access	Mnemonic	Reset	Definition
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.
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 357 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 357. **Note:** This bit is ignored if PADEN is set to 0.

Table 211: MAC Configuration Register #2

Bits	Access	Mnemonic	Reset	Definition
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 357.
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.
D02	R/W	HUGE	0	**Huge frame enable** 0 Transmit and receive frames are limited to the MAXF value in the Maximum Frame register. 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.

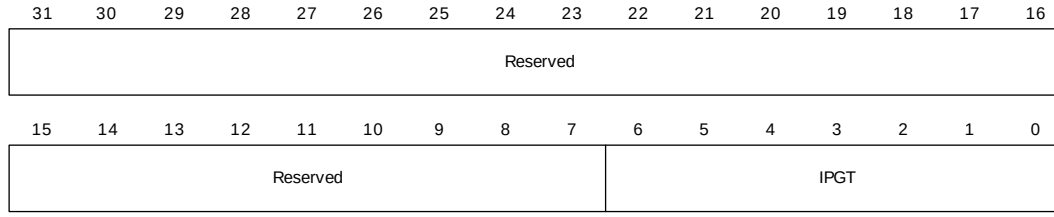
Table 211: MAC Configuration Register #2

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



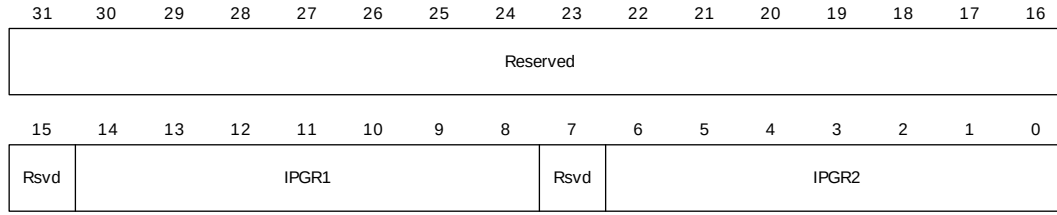
Register bit assignment

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 212: Back-to-Back Inter-Packet-Gap register

Non Back-to-Back Inter-Packet-Gap register

Address: A060 040C



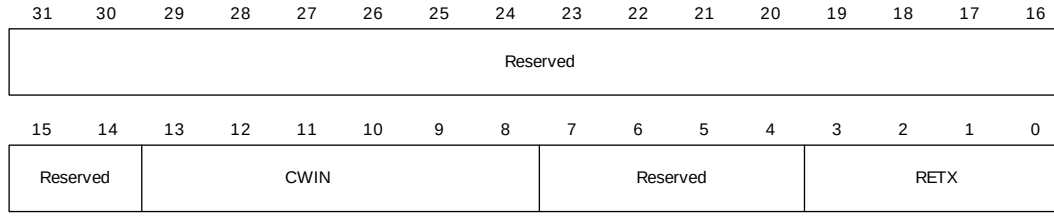
Register bit assignment

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 213: Non Back-to-Back Inter-Packet-Gap register

Collision Window/Retry register

Address: A060 0410



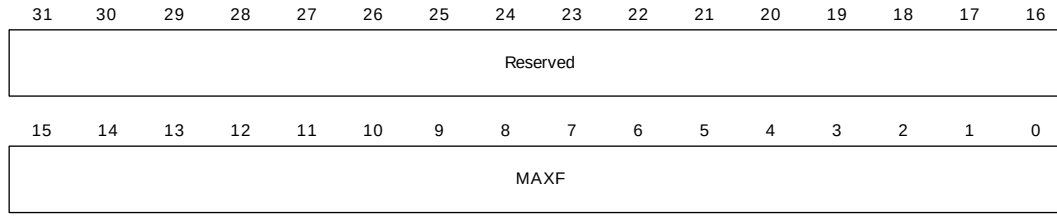
Register bit assignment

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 214: Collision Window/Retry register

Maximum Frame register

Address: A060 0414



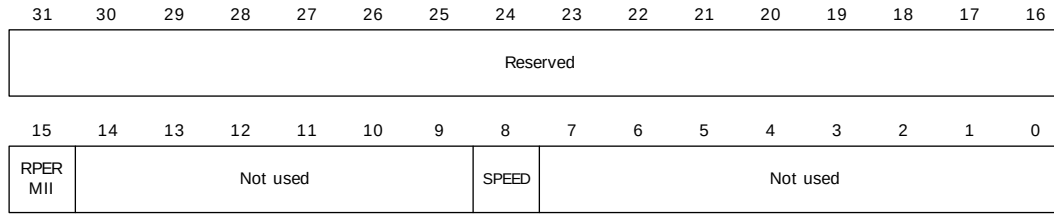
Register bit assignment

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 215: Maximum Frame register

PHY Support register

Address: A060 0418



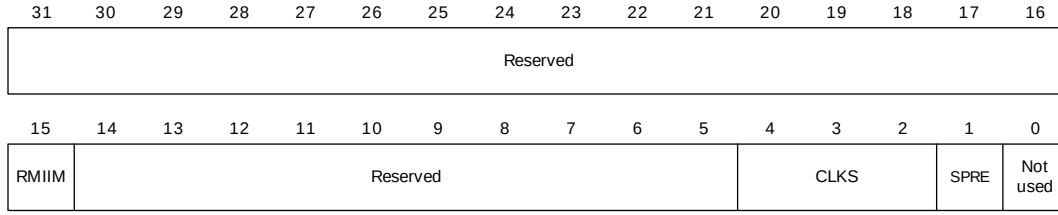
Register bit assignment

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 216: 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. See the "Clocks field settings" table for settings that can be used with the sample AHB clock (that is, hclk) frequencies shown in Table 3, “Sample clock frequency settings with 29.4912 MHz crystal,” on page 36.
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.

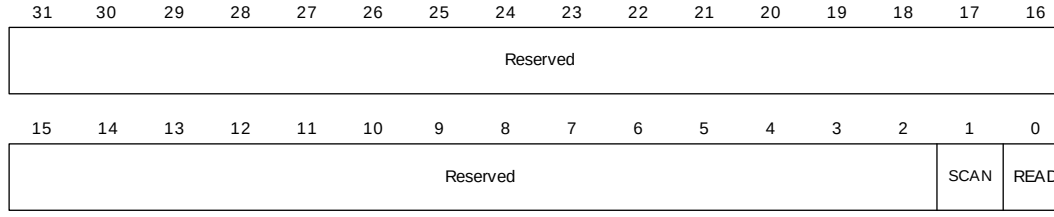
Table 217: MII Management Configuration register

Clocks field settings

CLKS field	Divisor	AHB bus clock for 2.5 MHz (max) MII management clock	AHB bus clock for 12.5 MHz (max) MII management clock
000	4		
001	4		
010	6		51.6 MHz
011	8		88.5 MHz / 77.4 MHz
100	10		
101	20		
110	30	51.6 MHz	
111	40	88.5 MHz / 77.4 MHz	

MII Management Command register

Address: A060 0424



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

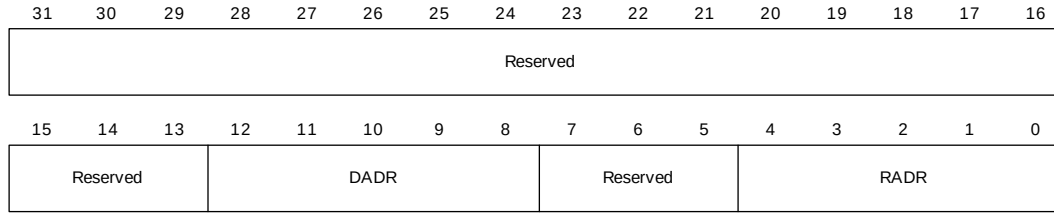
Table 218: MII Management Command register

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 218: MII Management Command register

MII Management Address register

Address: A060 0428



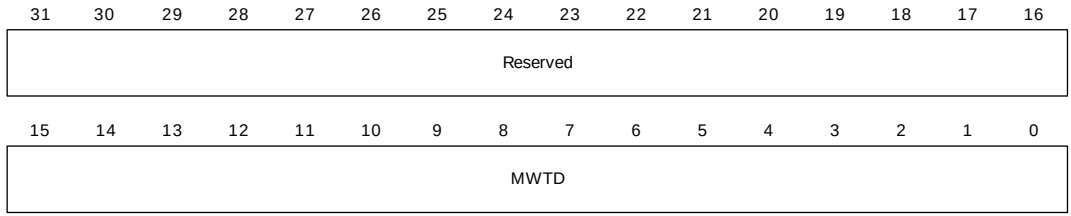
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	N/A	Reserved	N/A	N/A
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 219: MII Management Address register

MII Management Write Data register

Address: A060 042C



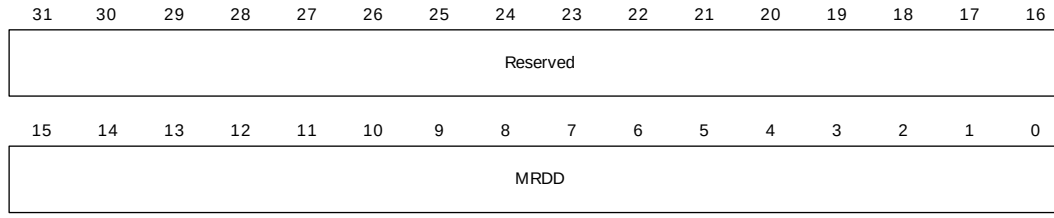
Register bit assignment

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. The write operation completes when the BUSY bit in the MII Management Indicators register returns to 0.

Table 220: MII Management Write Data register

MII Management Read Data register

Address: A060 0430



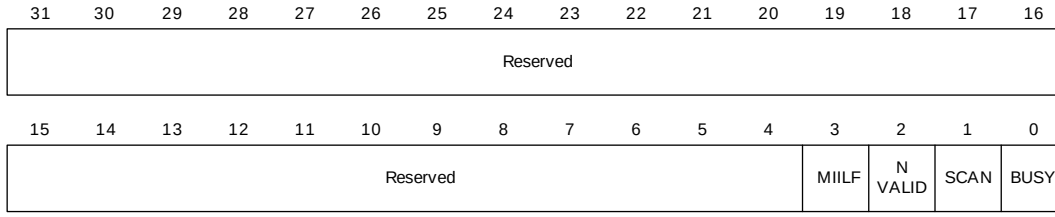
Register bit assignment

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. Read data is available after the BUSY bit in the MII Management Indicators register returns to 0.

Table 221: MII Management Read Data register

MII Management Indicators register

Address: A060 0434



Register bit assignment

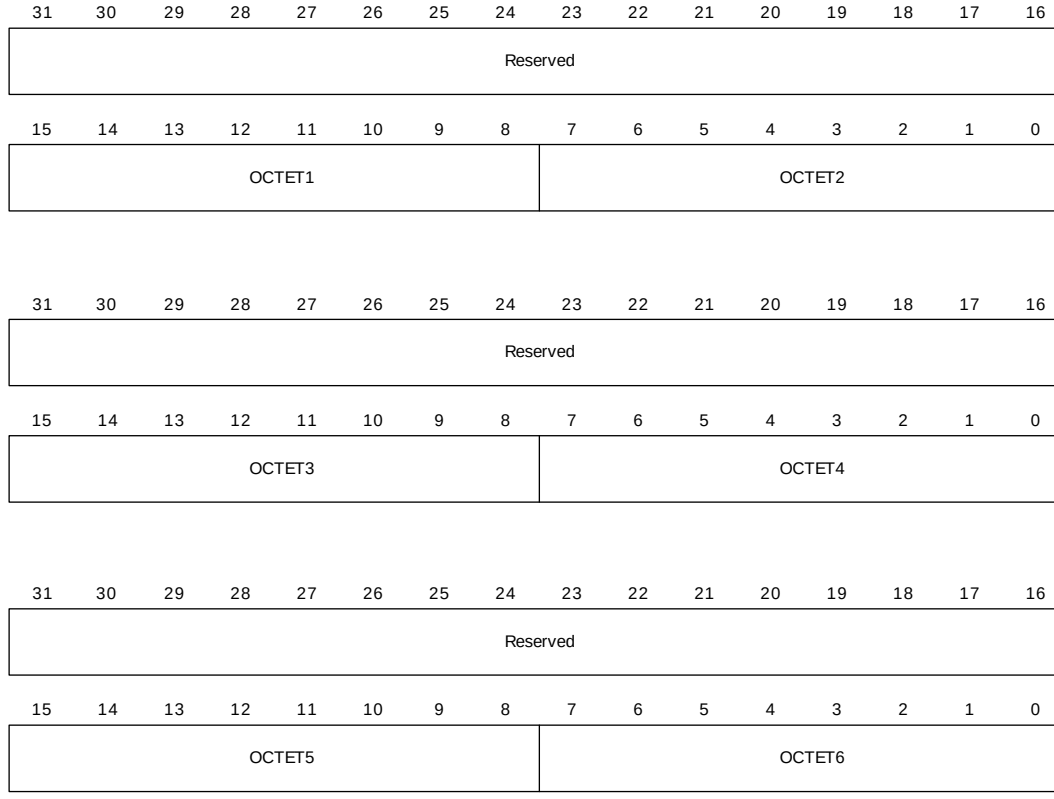
Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	MIILF	0	**MII link failure** When set to 1, indicates that the PHY currently has a link fail condition.
D02	R	NVALID	0	**Read data not valid** When set to 1, indicates that the MII Management read cycle has not completed and the 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 222: 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 324).



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

Table 223: Station Address registers

Bits	Access	Mnemonic	Reset	Description
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 223: Station Address registers

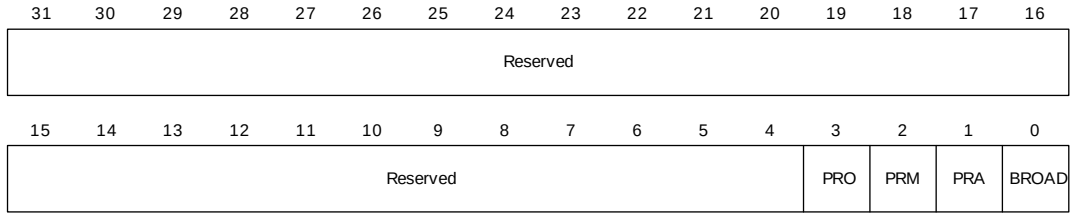
Note: Octet #6 is the first byte of a frame received from the MAC. Octet #1 is the last byte of the station address received from the MAC.

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 324).

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 224: Station Address Filter register

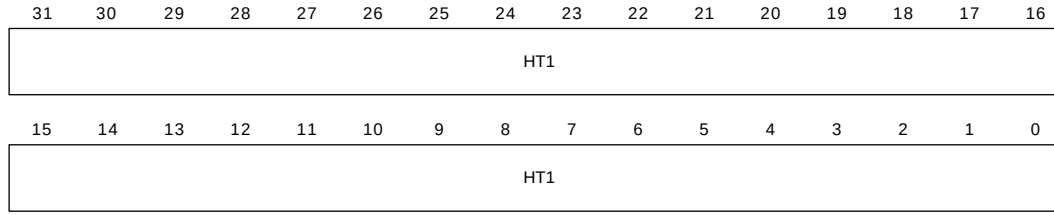
Register Hash Tables

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



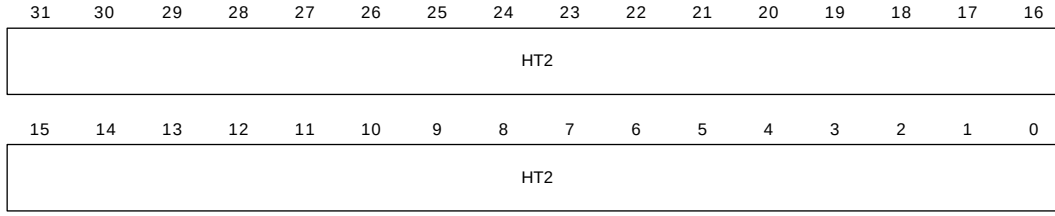
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	HT1	0x00000000	CRC 31:00

Table 225: Hash Table Register 1

HT2

Address: A060 0508



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	HT2	0x00000000	CRC 63:32

Table 226: 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 227, 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 227: 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_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 228: 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

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 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

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

Table 229: Transmit statistics counters address map

Address	Register	Transmit counters	R/W
A060_0728	TUND	Transmit undersize frame counter	R/W
A060_072C	TFRG	Transmit fragments frame counter	R/W

Table 229: 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 1532 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 and the MAC is not checking the FCS, 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 230 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

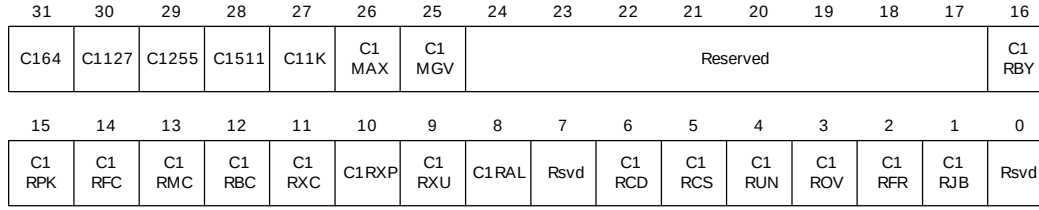
Table 230: General Statistics register address map

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 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



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

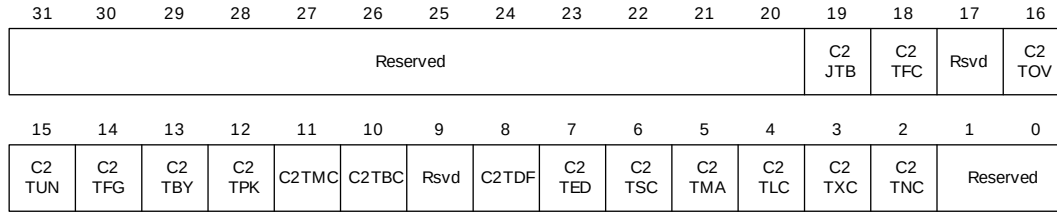
Table 231: Carry Register 1

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 231: Carry Register 1

Carry Register 2

Address: A060 0734



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

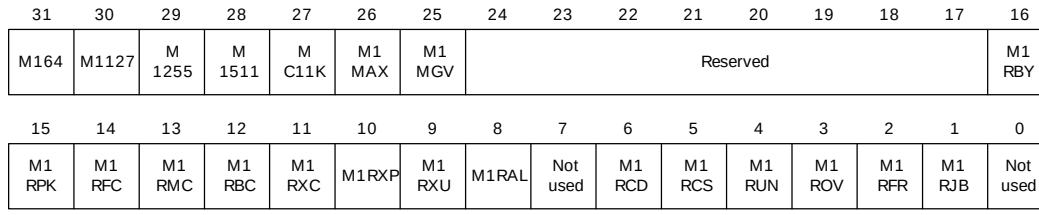
Table 232: Carry Register 2

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 232: Carry Register 2

Carry Register 1 Mask register

Address: A060 0738



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

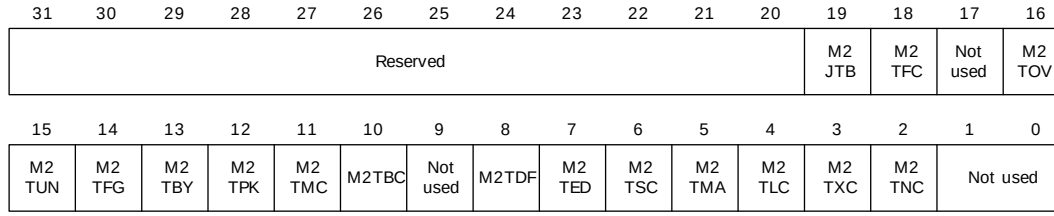
Table 233: Carry Register 1 Mask register

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	R/W	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 233: Carry Register 1 Mask register

Carry Register 2 Mask register

Address: A060 073C

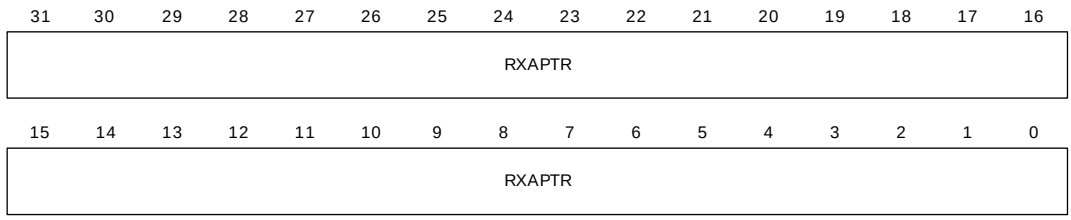


Bits	Access	Mnemonic	Reset	Description
D31:20	N/A	Reserved	N/A	N/A
D19	R/W	M2TJB	1	Mask register 2 TJBR counter carry bit mask
D18	R/W	M2TFC	1	Mask register 2 TFCS counter carry bit mask
D17	R/W	Not used	1	Always write as 1.
D16	R/W	M2TOV	1	Mask register 2 TOVR counter carry bit mask
D15	R/W	M2TUN	1	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 234: Carry Register 2 Mask register

RX_A Buffer Descriptor Pointer register

Address: A060 0A00



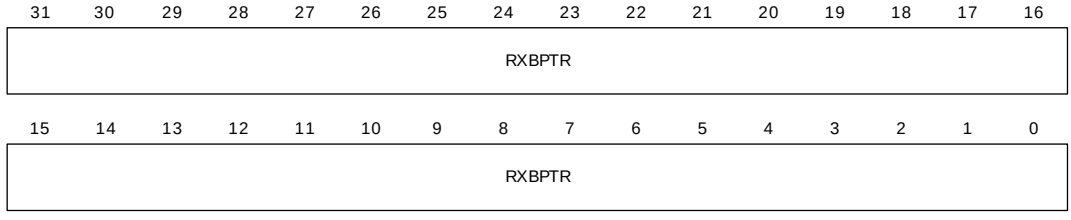
Register bit assignment

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 235: RX_A Buffer Descriptor Pointer register

RX_B Buffer Descriptor Pointer register

Address: A060 0A04



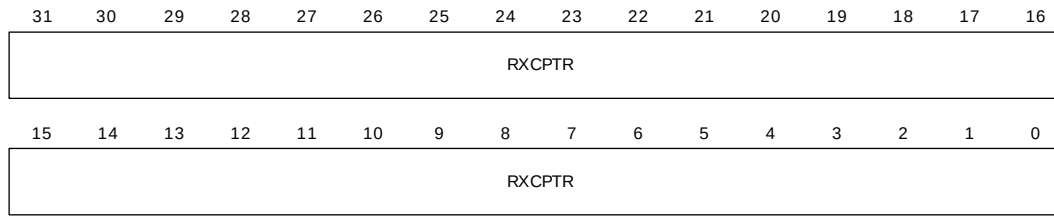
Register bit assignment

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 236: RX_B Buffer Descriptor Pointer register

RX_C Buffer Descriptor Pointer register

Address: A060 0A08



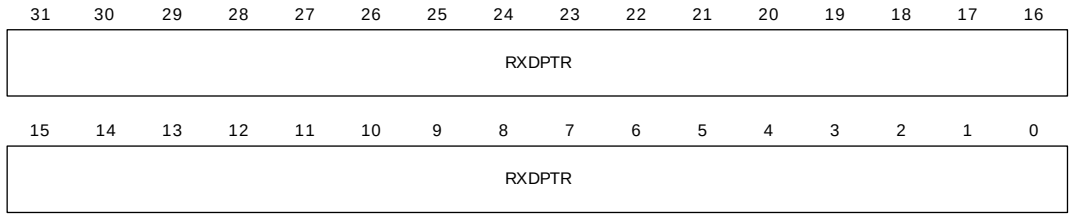
Register bit assignment

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 237: 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 238: RX_D Buffer Descriptor Pointer register

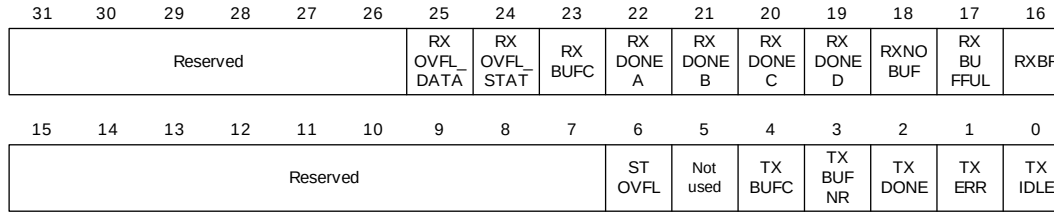
Ethernet Interrupt Status register

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.



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 #1 when an overflow condition occurs.
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.

Table 239: Ethernet Interrupt Status register

Bits	Access	Mnemonic	Reset	Description
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
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	Last frame not transmitted successfully. Assigned to TX interrupt. See "Ethernet Interrupt Status register" on page 391 for information about restarting the transmitter when this bit is set.

Table 239: Ethernet Interrupt Status register

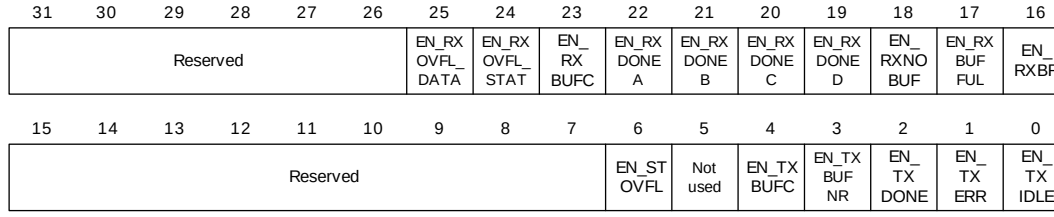
Bits	Access	Mnemonic	Reset	Description
D00	R/C	TXIDLE	0	TX_WR logic has no frame to transmit. Assigned to TX interrupt. See "Ethernet Interrupt Status register" on page 391 for information about restarting the transmitter when this bit is set.

Table 239: 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.



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.

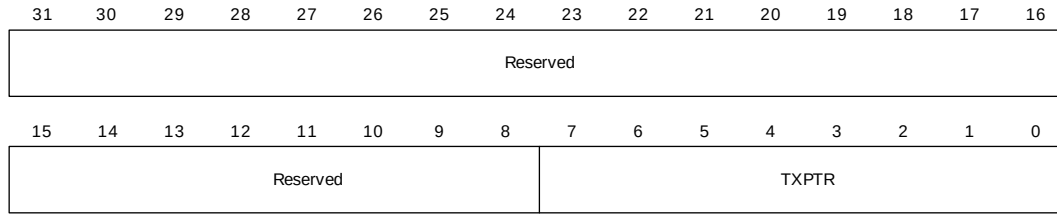
Table 240: Ethernet Interrupt Enable register

Bits	Access	Mnemonic	Reset	Description
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 240: Ethernet Interrupt Enable register

TX Buffer Descriptor Pointer register

Address: A060 0A18



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A

Table 241: TX Buffer Descriptor Pointer register

Bits	Access	Mnemonic	Reset	Description
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 241: TX Buffer Descriptor Pointer register

Transmit Recover Buffer Descriptor Pointer register

Address: A060 0A1C

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A

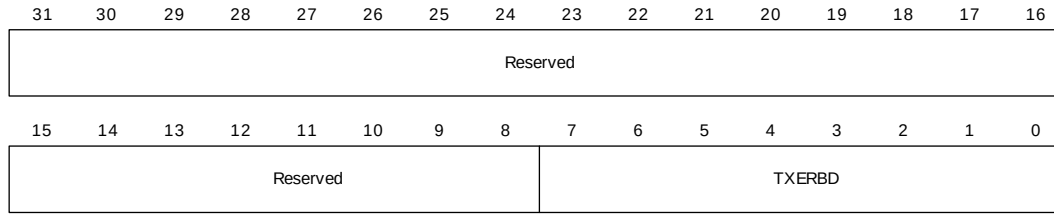
Table 242: Transmit Recover Buffer Descriptor Pointer register

Bits	Access	Mnemonic	Reset	Description
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.

Table 242: 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

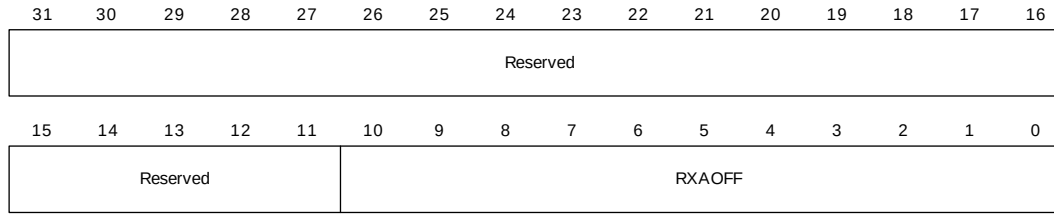
Table 243: TX Error Buffer Descriptor Pointer register

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 243: TX Error Buffer Descriptor Pointer register

RX_A Buffer Descriptor Pointer Offset register

Address: A060 0A28



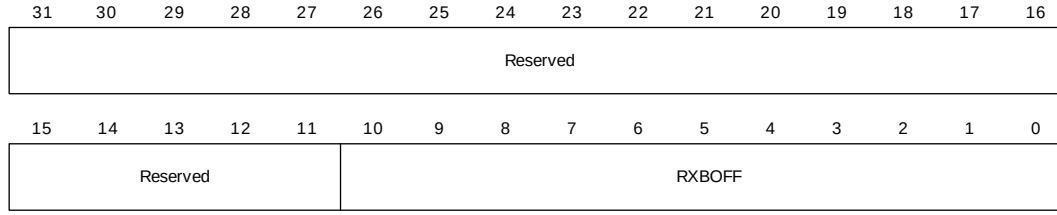
Register bit assignment

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 244: RX_A Buffer Descriptor Pointer Offset register

RX_B Buffer Descriptor Pointer Offset register

Address: A060 0A2C



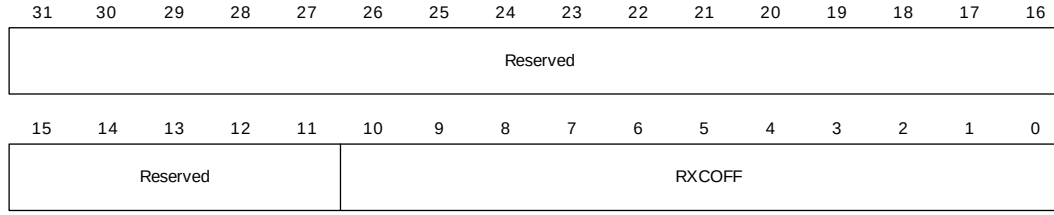
Register bit assignment

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 245: RX_B Buffer Descriptor Pointer Offset register

RX_C Buffer Descriptor Pointer Offset register

Address: A060 0A30



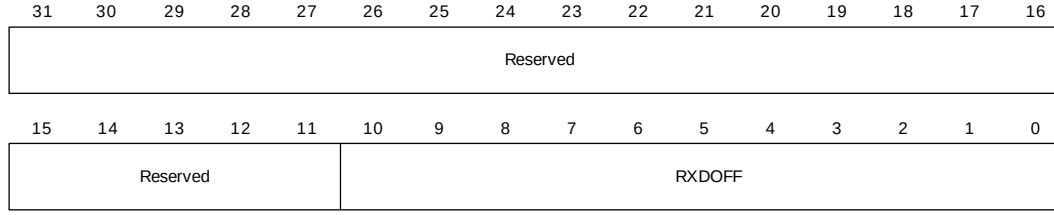
Register bit assignment

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 246: RX_C Buffer Descriptor Pointer Offset register

RX_D Buffer Descriptor Pointer Offset register

Address: A060 0A34



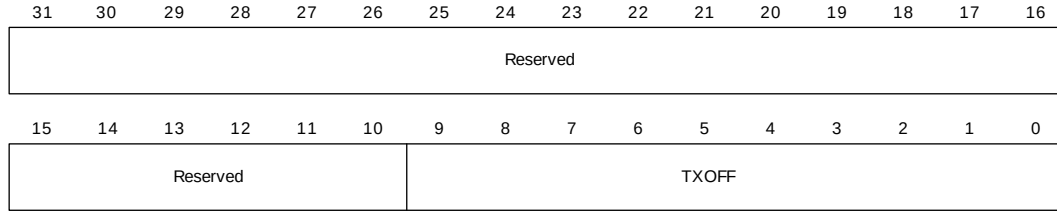
Register bit assignment

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 247: RX_D Buffer Descriptor Pointer Offset register

Transmit Buffer Descriptor Pointer Offset register

Address: A060 0A38



Register bit assignment

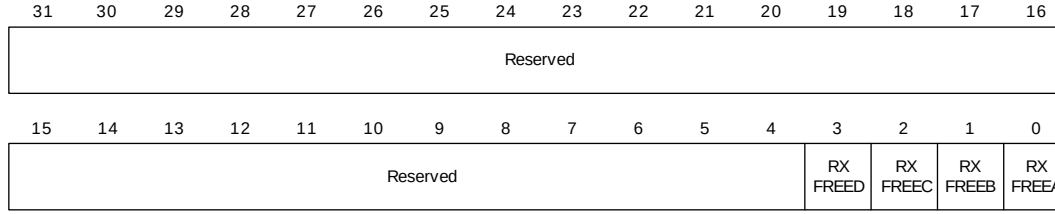
Bits	Access	Mnemonic	Reset	Description
D31:10	N/A	Reserved	N/A	N/A
D09:00	R	TXOFF	0x000	Contains a 10-bit byte offset from the start of the transmit ring in the TX buffer descriptor RAM. The offset is updated at the end of the TX packet, and will have the offset to the next buffer descriptor that will be used. TXOFF can be used to determine from where the TX_WR logic will grab the next packet.

Table 248: 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 249: 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 64, "Transmit buffer descriptor format," on page 331, 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;

}
```

BBus Bridge

CHAPTER 7

The NS9360 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 BBus bridge connects the AMBA AHB bus to the proprietary Digi BBus. Both bus interfaces have a master and a slave interface.

BBus bridge functions

The primary function of the BBus bridge is to connect the AHB bus to the proprietary Digi bus. The BBus bridge also performs these “secondary” 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.
- **Data buffer.** The data buffer is in the datapath between the BBus masters and the external memory, and allows the system to perform octal word fetches from memory.
- **Memory subsystem.** The BBus bridge serves as a 64-byte memory subsystem located at offset 0xFFFF_0000. This memory space allows software to relocate the interrupt vectors, which can be useful during code development.

The next figure shows the BBus bridge functions in relation to the AHB bus and the BBus.

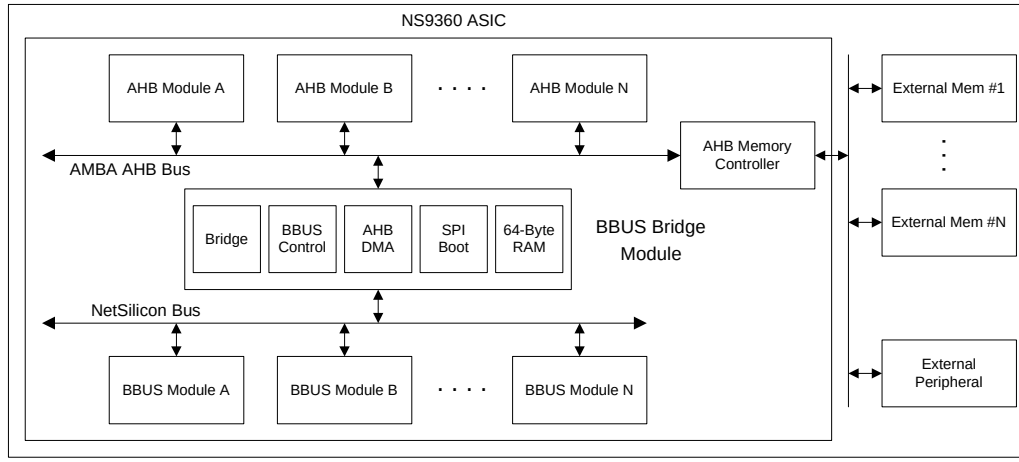


Figure 78: Basic block diagram

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 90 MHz; the BBus operates at half the AHB clock frequency.

- The reference clocks for each bus are asynchronous.
- A 4-entry FIFO is implemented in the BBus-to-AHB data path to allow burst transfers.
- A 32-entry prefetch buffer in the BBus-to-AHB datapath allows the hardware to prefetch data for the BBus masters.
- The AHB-to-BBus data path does not support burst transfers, allowing only a single entry FIFO to be implemented in the data path.

Figure 79 shows BBus bridge control logic.

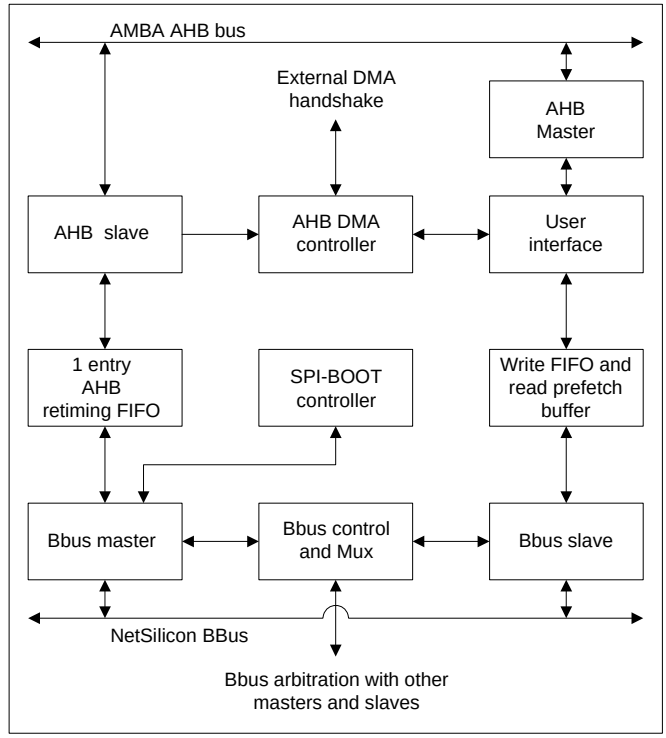


Figure 79: BBus bridge block diagram

DMA accesses

There are two DMA controllers on the NS9360 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.

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 master interfaces, an IEEE 1284 interface, a full-speed USB Host and a full-speed USB device.

- The SPI master interface supports a maximum of 11.25 Mbps of full-duplex traffic.
- The IEEE 1284 interface supports a maximum of 2 Mbps.
- The USB Device and USB Host support a maximum of 12 Mbps of half-duplex traffic.

The total peripheral bandwidth is 71 Mbps. Adding 9 Mbps for DMA buffer descriptors brings the total requirement to 10 MBps, less than on AHB timeslot with a full 8-slot rotation.

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 four 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.

The next table 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 Device		Y

Table 304: BBus master and slave modules

Module	Master	Slave
USB DMA		Y
USB Host	Y	Y

Table 304: 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.

The next table specifies the address space given to each peripheral.

Base address	Peripheral
0x9000 0000	BBus DMA controller
0x9010 0000	Reserved
0x9020 0000	SER Port #B
0x9020 0040	SER Port #A
0x9030 0000	SER Port #C
0x9030 0040	SER Port #D

Table 305: BBus peripheral address map

Base address	Peripheral
0x9040 0000	IEEE1284 controller
0x9050 0000	I^2C controller
0x9060 0000	BBus utility
0x9070 0000	Real Time Clock
0x9080 0000	USB Host
0x9090 0000	USB Device

Table 305: BBus peripheral address map

64 byte memory space

There is a 64 byte memory space starting at address 0xFFFF 0000 and ending at 0xFFFF 003F. The memory is organized as 16 32-bit words. Each word can be accessed with zero wait states. Byte or halfword accesses to this memory space are not allowed.

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. 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 431).
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 431).

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 the NS9360 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 "DMA Channel 1/2 Buffer Descriptor Pointer register" on page 429). 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 with 16 bytes. Configuring the DMA channel for more than the maximum number of buffer descriptors results in unpredictable behavior.

Figure 80 shows the DMA buffer descriptor. Table 306 describes each section.

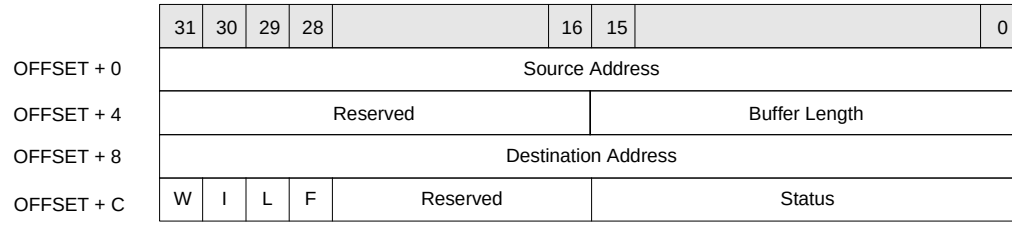


Figure 80: BBus bridge DMA buffer descriptor

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. This is useful for debugging error conditions or for determining the number of bytes transferred before the DONE signal was asserted.
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.
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.

Table 306: BBus bridge DMA buffer descriptor definition

Field/Section	Description
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. Use this bit when multiple descriptors are chained together to make up a data frame.
F	The **full** bit. When set, this bit indicates that the buffer is full. A DMA channel clears this bit after completing the transfer, but does not attempt a transfer with the F bit clear. When firmware modifies the F bit, the firmware must also write a 1 to the CE bit in the DMA Channel Control register to activate the idle channel.
Reserved	Write zero to this field.
Status	Not used. Read back 0x0000.

Table 306: 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 435) and returns to the idle state after fetching a buffer descriptor with the F bit in the incorrect state.

A DMA channel always closes the current descriptor and moves on to the next descriptor when a DMA transfer is terminated by assertion of the DONE signal.

Peripheral DMA read access

Figure 81 and Figure 82 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 AHB CLK period. The delay from the falling edge of CS# to the rising edge of READ_EN is always equal to one AHB 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 315).

The NS9360 treats DMA read accesses from an external peripheral as asynchronous operations. 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 AHB CLK cycle period

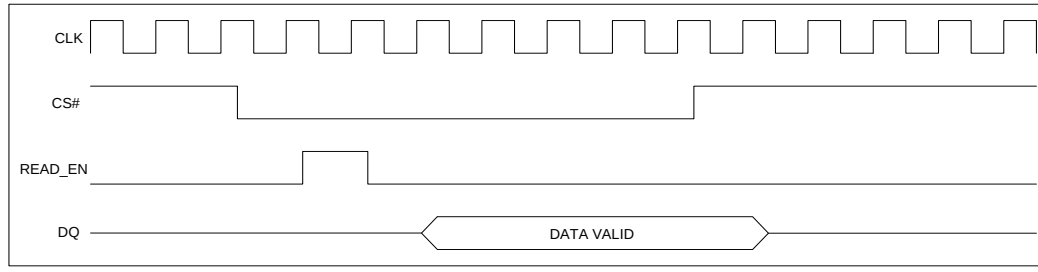


Figure 81: Peripheral DMA single read access

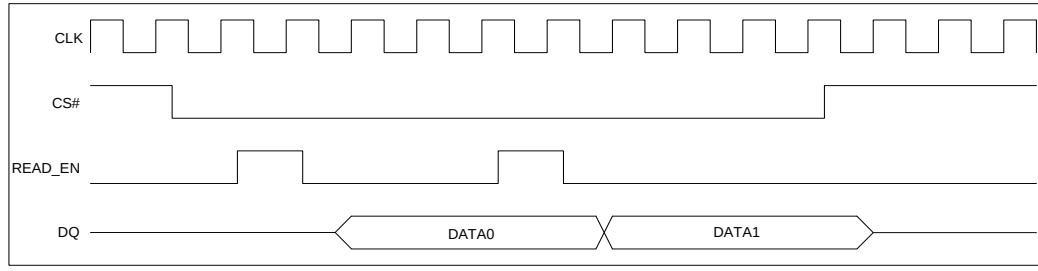


Figure 82: Peripheral DMA burst read access

Peripheral DMA write access

Figure 83 and Figure 84 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 AHB 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 Write Delay 0–3 registers" on page 317 and "Static Memory Write Enable Delay 0–3 registers" on page 313).

Note that the READ_EN signal is not used during peripheral DMA write accesses.

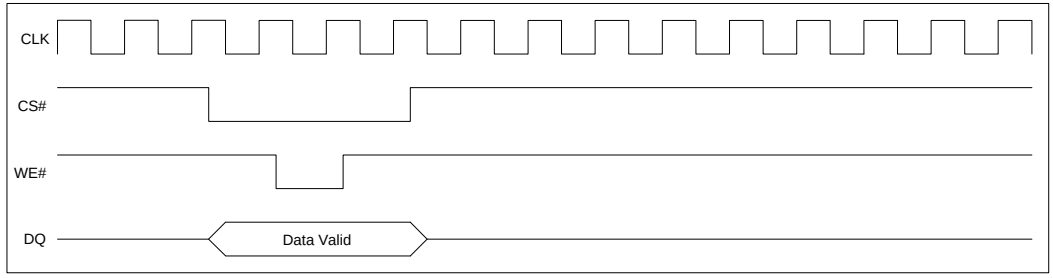


Figure 83: Peripheral DMA single write access

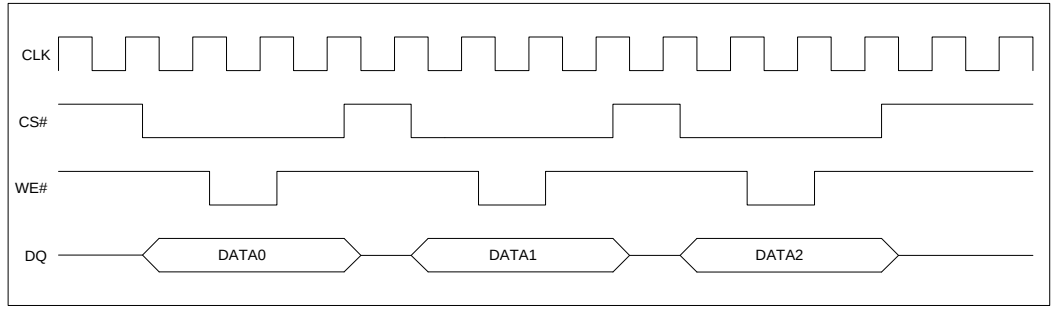


Figure 84: Peripheral DMA burst write access

Peripheral REQ and DONE signalling

The NS9360 treats the REQ and DONE signals as asynchronous level signals. The external peripheral can initiate a DMA transfer at any time by asserting the REQ signal. The external peripheral can pause the DMA transfer at any time by deasserting the REQ signal. The REQ signal can be deasserted during a transfer but, if the peripheral is configured for burst access, the burst will complete. The DMA transfer control logic will remain paused until the REQ signal is reasserted.

The external peripheral can terminate the DMA transfer at any time by asserting the DONE signal. The peripheral must also deassert the REQ signal when it asserts the DONE signal. The DONE signal can be asserted during a transfer but, if the peripheral is configured for burst access, the burst will complete. When it finds a DONE assertion, the DMA control logic closes the current buffer descriptor, asserts a premature buffer completion status, and pauses until the REQ signal is reasserted. The DONE signal must be deasserted no later than four AHB clock cycles before reasserting the REQ signal.

The REQ and DONE signals are ignored for memory-to-memory DMA transfers that are initiated by software writing a 1 to the CG field in the DMA Control register.

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 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 82us. Response to the REQ signal deassertion is immediate, as that function does not require any system bandwidth.

- The chip selects that are programmed for use by an external DMA channel must not be used for any other purposes.

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 * 8 * 2) / 177 MHz) = 1.45us

This AHB access pattern looks like this:

Cpu, Erx, Cpu, Etx; Cpu, Lcd, Cpu, *BBus*

Best case:

AHB access = ((16 * 4 * 2) / 177 MHz) = 0.72us

This AHB access pattern looks like this:

Cpu, Erx, Cpu, *BBus*; Cpu, Etx, Cpu, *BBus*; Cpu, Lcd, 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, USB Host,

and IEEE 1284 are half-duplex-only peripherals, so only the more “pessimistic” 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 = 1.45us * ((#Receive * 4) + (#Transmit * 5) + 6)

Best case latency = 0.72us * ((#Receive * 4) + (#Transmit * 5) + 6)

In a fully loaded system with four UARTs, IEEE 1284, USB Device, and USB Host, the worst case latency is 82us and the best case latency is 41us.

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 438 for more information.)

Important: The interrupt(s) must be serviced in the peripheral in which the interrupt(s) originated.

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. See the bootstrap initialization section in the System Control Module chapter for 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 307: NS9360 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 the SPI-EEPROM to external memory.

The SPI boot operates in SPI mode 0. The SPI clock frequency is fixed at the CPU clock divided by 128. See the SPI mode 0 timing diagrams in the Timing chapter for details.

Important:

- SPI-EEPROM must be connected to SER port B; the boot logic does not communicate with any other SER port.
- The endianess of the image in SPI must match the endianess of the system.

Calculation and example

This equation calculates the amount of time, in seconds, required to copy the contents of the SPI-EEPROM to external memory:

$$\text{Time} = (1 / \text{freq}) * \text{EEPROM}_{\text{SIZE}}$$

Example

SPI master clock frequency = 1.4 MHz

SPI-EEPROM = 256 Kb

Time for operation to complete = 183 ms

Serial Channel B configuration

When exiting 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 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 308: SPI master mode boot configuration

Memory Controller configuration

The memory controller exits the reset state in non-operational mode, which 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. The next table 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: 4-(1+2) = 1 The result is a pad entry length of 1 byte.
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: ((Code image size in bytes + 128) / 4) + 1
SDRAM config	All SDRAM components contain a Mode register. This register contains 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 such that it is aligned to the row address bits as specified in "Address mapping," beginning on page 243. For example, 4Mx16 components can be combined to create a 32-bit bus. These parts require 12 row address bits. With 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 0x0000002B given a 88 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.

Table 309: ARM boot configuration

EEPROM entry	Description
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.
DynamicConfig0	See the Memory Controller chapter. 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.
DynamicRasCas0	See the Memory Controller chapter.
Reserved	The remaining bytes are undefined. The final byte address of the 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 309: ARM boot configuration

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. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

This table lists the configuration and status registers in the BBus Bridge module.

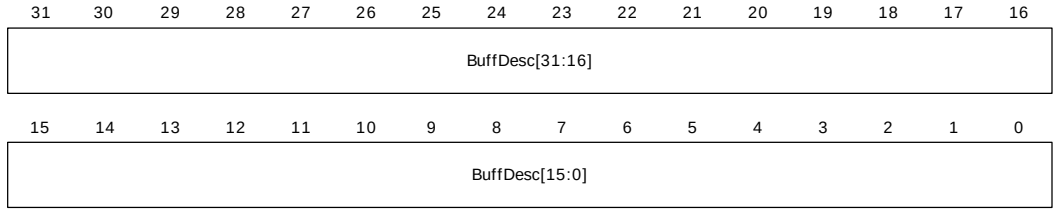
Address	Register
A040 0000	DMA Channel 1 Buffer Descriptor Pointer
A040 0004	DMA Channel 1 Control register
A040 0008	DMA Channel 1 Status and Interrupt Enable
A040 000C	DMA Channel 1 Peripheral Chip Select
A040 0020	DMA Channel 2 Buffer Descriptor Pointer
A040 0024	DMA Channel 2 Control register
A040 0028	DMA Channel 2 Status and Interrupt Enable
A040 002C	DMA Channel 2 Peripheral Chip Select
A040 1000	BBus Bridge Interrupt Status
A040 1004	BBus Bridge Interrupt Enable
A040 1008	BBus Bridge Prefetch (Burst-8) Buffer Enable

Table 310: BBus Bridge Control and Status registers

DMA Channel 1/2 Buffer Descriptor Pointer register

Address: A040 0000 / 0020

The DMA Channel 1/2 Buffer Descriptor Pointer 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.



Register bit assignment

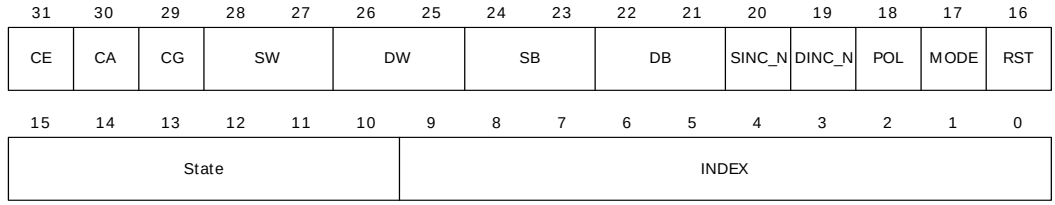
Bit(s)	Access	Mnemonic	Reset	Description
D31:00	RW	BuffDesc	0x00000000	**Buffer descriptor** 32-bit pointer to a buffer descriptor.

Table 311: DMA Channel 1/2 Buffer Descriptor Pointer register bit definition

DMA Channel 1/2 Control register

Address: A040 0004 / 0024

The DMA Channel 1/2 Control register contains required DMA transfer control information. The BBus bridge contains a DMA Channel Control register for each channel.



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 to initiate additional DMA transfers after a DMA channel has entered the IDLE state for any reason.
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, to allow software to initiate a memory-to-memory DMA transfer. The dma_req and dma_done signals are not used during memory-to-memory transfers.

Table 312: DMA Channel 1/2 Control register bit definition

Bit(s)	Access	Mnemonic	Reset	Description
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
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.
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. Used by the DMA controller in all modes when referring to a memory address.

Table 312: DMA Channel 1/2 Control register bit definition

Bit(s)	Access	Mnemonic	Reset	Description
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. Used by the DMA controller in all modes when referring to a memory address.
D18	R/W	POL	0	**Control signal polarity** 0 Active high signals 1 Active low signals Defines the active polarity of the dma_req and dma_done signals.
D17	R/W	MODE	0	**Fly-by mode** 0 Peripheral-to-memory fly-by-write DMA transfer 1 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 those DMA transfers initiated by writing a 1 to the CG field in this register.
D16	R/W	RST	0	**Reset** Forces a reset of the DMA channel. Writing a 1 to this field forces all fields in this register, *except the index field*, to the reset state. The index field is written with the value specified on signals HWDATA[9:0]. Writing a 1 to this field while the DMA channel is operational can have unpredictable results.

Table 312: DMA Channel 1/2 Control register bit definition

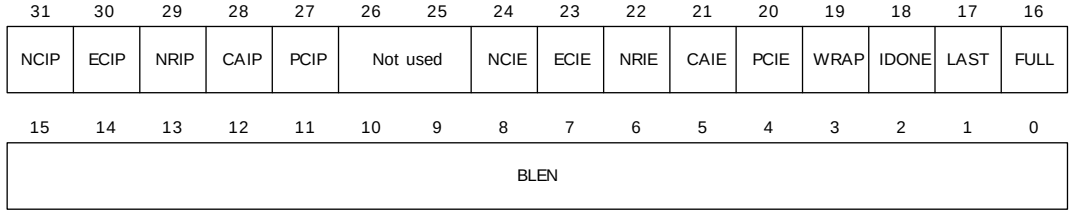
Bit(s)	Access	Mnemonic	Reset	Description
D15:10	R	STATE	0	**State** Contains the current value of the DMA controller state machine; 0x00 = IDLE 0x04 = FETCH_BD1 0x05 = FETCH_BD2 0x06 = FETCH_BD3 0x08 = UPDATE_BD1 0x09 = UPDATE_BD2 0x0a = UPDATE_BD3 0x10 = READ_DAT1 0x11 = READ_DAT2 0x12 = READ_DAT3 0x14 = READ_DAT4 0x15 = READ_DAT5 0x16 = READ_DAT6 0x17 = READ_DAT7 0x20 = WRITE_DAT1 0x21 = WRITE_DAT2 0x22 = WRITE_DAT3
D09:00	R	INDEX	0x00000000	**Index** Identifies the current byte offset pointer relative to the buffer descriptor pointer. This field can be written only when the RST field is written to a 1.

Table 312: DMA Channel 1/2 Control register bit definition

DMA Status and Interrupt Enable register

Address: A040 0008 / 0028

This 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 appropriate DMA Channel Control register to abort the current buffer descriptor and go to the next buffer descriptor.

Table 313: DMA Status and Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
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 F bit is not set, the descriptor is considered invalid and the NRIP field is set. When the NRIP bit 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. **Note:** 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.
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	2'b00	Always set this field to 0.
D24	R/W	NCIE	0x0	**Enable NCIP interrupt generation**
D23	R/W	ECIE	0x0	**Enable ECIP interrupt generation** This bit should always be enabled during normal operation.
D22	R/W	NRIE	0x0	**Enable NRIP interrupt generation**
D21	R/W	CAIE	0x0	**Enable CAIP interrupt generation** This bit should always be enabled 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.

Table 313: DMA Status and Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
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.

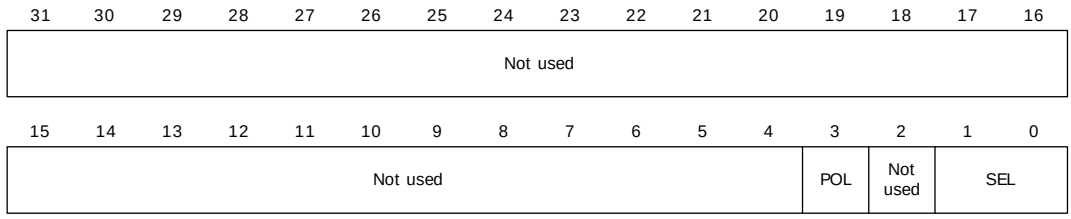
Table 313: DMA Status and Interrupt Enable register bit definition

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.

Important: The chip selects that are programmed for use by an external DMA channel must not be used for any other purpose.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	Not used	0	Always set to 0.

Table 314: DMA Peripheral Chip Select register

Bits	Access	Mnemonic	Reset	Description
D03	R/W	POL	0	**Chip select polarity** 0 Active high signal 1 Active low signal Defines the polarity of the memory interface chip select signal (stcsout[n]_n) connected to the external peripheral.
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: 0 stcsout[0] 1 stcsout[1] 2 stcsout[2] 3 stcsout[3]

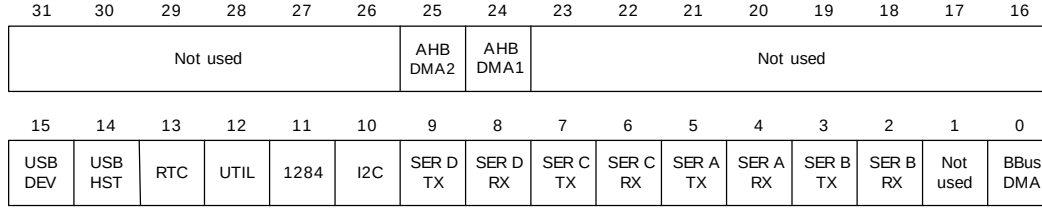
Table 314: DMA Peripheral Chip Select register

BBus Bridge Interrupt Status register

Address: A040 1000

The BBus Bridge Interrupt Status register contains the interrupt status of the BBus peripherals. All interrupts must be serviced in the originating module.

- Reading a 1 indicates that an interrupt is pending.
- Reading a 0 indicates that no interrupt is pending.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	R	Not used	0	Always set this field to 0.
D25	R	AHB_DMA2	0	AHB DMA channel #2 has asserted its interrupt.
D24	R	AHB_DMA1	0	AHB DMA channel #1 has asserted its interrupt.
D23:16	R	Not used	0	Always set this field to 0.
D15	R	USBDEV	0	USB Device module has asserted its interrupt.
D14	R	USBHST	0	USB Host module has asserted its interrupt.
D13	R	RTC	0	Real time clock module has asserted its interrupt.
D12	R	UTIL	0	BBus Utility module has asserted its interrupt.
D11	R	1284	0	IEEE-1284 module has asserted its interrupt.
D10	R	I2C	0	I^2C module has asserted its interrupt.
D09	R	SER_D_TX	0	SER transmit module D has asserted its interrupt.
D08	R	SER_D_RX	0	SER receive module D has asserted its interrupt.
D07	R	SER_C_TX	0	SER transmit module C has asserted its interrupt.
D06	R	SER_C_RX	0	SER receive module C has asserted its interrupt.
D05	R	SER_A_TX	0	SER transmit module A has asserted its interrupt.
D04	R	SER_A_RX	0	SER receive module A has asserted its interrupt.
D03	R	SER_B_TX	0	SER transmit module B has asserted its interrupt.
D02	R	SER_B_RX	0	SER receive module B has asserted its interrupt.
D01	R	Not used	0	Always set to 0.
D00	R	BBUS_DMA	0	BBus DMA module has asserted its interrupt.

Table 315: BBus Bridge Interrupt Status register

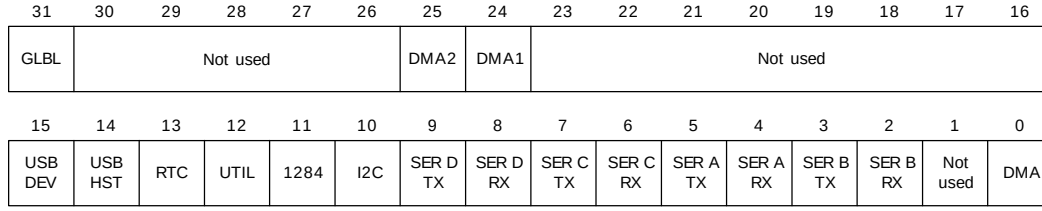
BBus Bridge Interrupt Enable register

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 affect the value on the bbus_int pin. Writing a 0 to a bit disables the interrupt.

These fields affect only the generation of the bbus_int signal; they do not affect the originating modules.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	GLBL	0	Enable bbus_int signal to propagate to the System Control module.
D30:26	R/W	Not used	0	Always set 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.
D23:16	R/W	Not used	0	Always set to 0.
D15	R/W	USBDEV	0	Enable interrupt from USB Device module.
D14	R/W	USBHST	0	Enable interrupt from USB Host module.
D13	R/W	RTC	0	Enable interrupt from RTC module.
D12	R/W	UTIL	0	Enable interrupt from BBus Utility module.
D11	R/W	1284	0	Enable interrupt from IEEE1284 module.
D10	R/W	I2C	0	Enable interrupt from I2C module.
D09	R/W	SER_D_TX	0	Enable interrupt from SER transmit module D.
D08	R/W	SER_D_RX	0	Enable interrupt from SER receive module D.
D07	R/W	SER_C_TX	0	Enable interrupt from SER transmit module C.

Table 316: BBus Bridge Interrupt Enable register bit definition

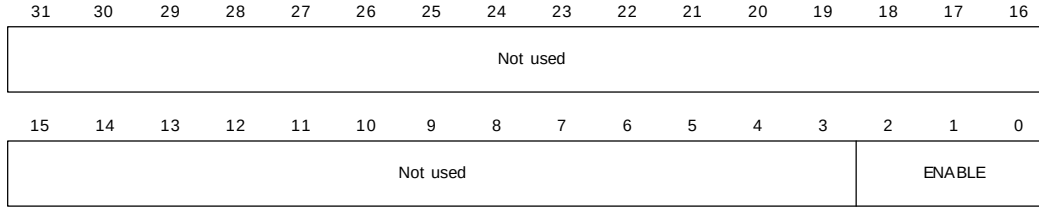
Bits	Access	Mnemonic	Reset	Description
D06	R/W	SER_C_RX	0	Enable interrupt from SER receive module C.
D05	R/W	SER_A_TX	0	Enable interrupt from SER transmit module A.
D04	R/W	SER_A_RX	0	Enable interrupt from SER receive module A.
D03	R/W	SER_B_TX	0	Enable interrupt from SER transmit module B.
D02	R/W	SER_B_RX	0	Enable interrupt from SER receive module B.
D01	R/W	Not used	0	Always set to 0.
D00	R/W	DMA	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 316: BBus Bridge Interrupt Enable register bit definition

BBus Bridge Prefetch (Burst-8) Buffer Enable register

Address: A040 1008

The BBus Bridge Prefetch (Burst-8) Buffer Enable register allows you to enable or disable the BBus bridge prefetch data buffer on a per-BBus-master basis.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:03	R	Not used	0	Always set to 0.

Table 317: BBus Bridge Prefetch (Burst-8) Buffer Enable register

Bits	Access	Mnemonic	Reset	Description
D02:00	R	ENABLE	0x7	**Buffer enable** Enables the prefetch data buffer in the BBus bridge on a per-BBus-master basis, allowing the BBus bridge to prefetch octal words from system memory. Writing a 0 to this field disables the feature, and the BBus bridge will fetch only data that is requested by the BBus masters: [2] USB Device DMA [1] USB Host [0] BBus DMA

Table 317: BBus Bridge Prefetch (Burst-8) Buffer Enable register

BBus DMA Controller

C H A P T E R 8

The NS9360 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 85 shows the data flow for fly-by DMA transfers.

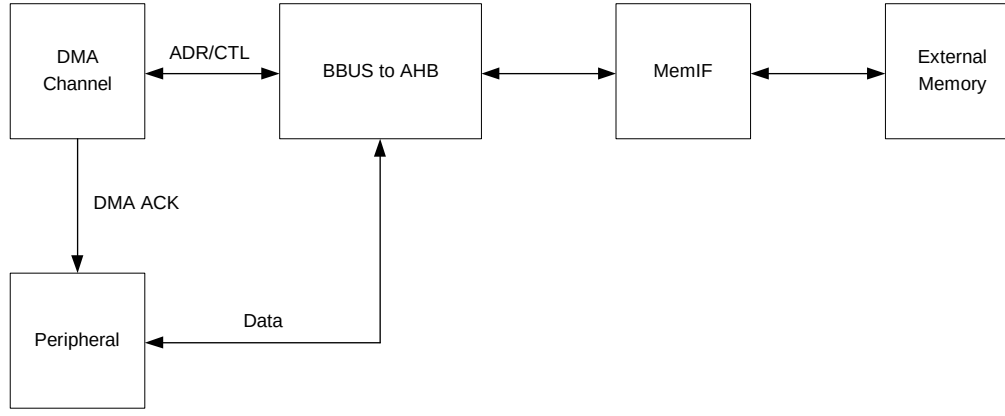


Figure 85: DMA fly-by transfers

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 86 shows the BBus DMA controller block.

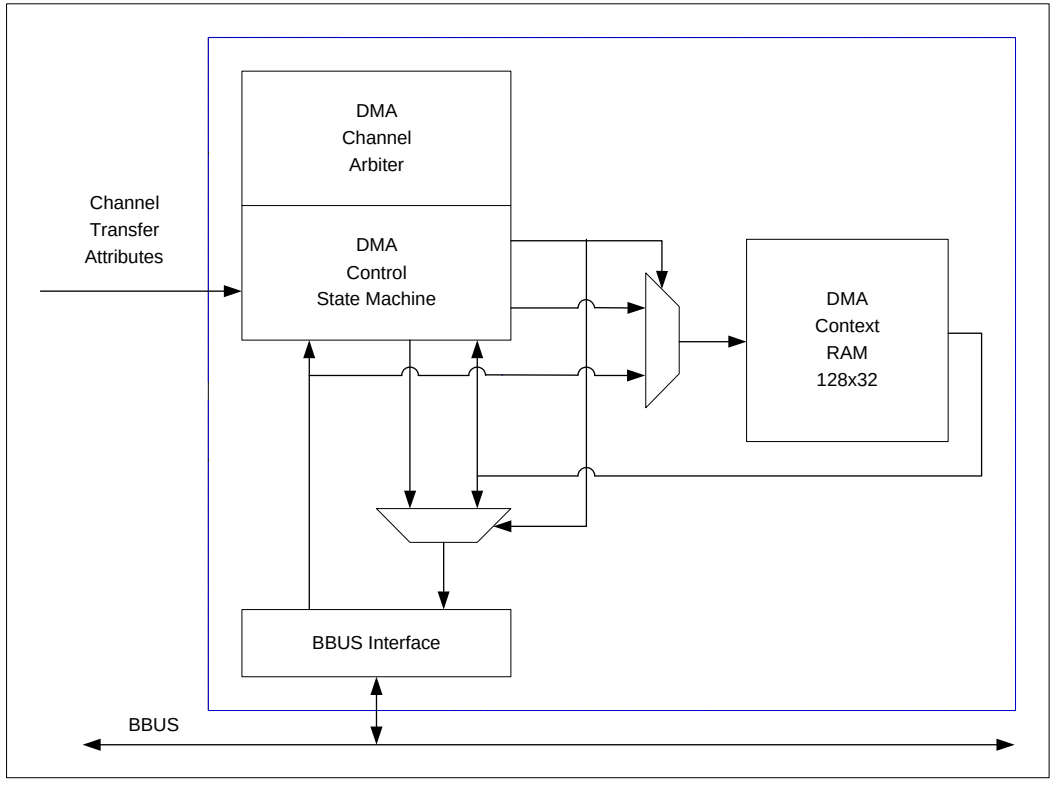


Figure 86: DMA controller block

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. The next table defines the entries that describe the state of each DMA channel.

Offset	Description
0x00	Buffer descriptor pointer
0x01	Control register
0x02	Status register
0x03	Not used
0x04	Source Address register
0x05	Buffer Length register
0x06	Destination Address register
0x07	Control flags and transfer status

Table 318: 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 87 shows the DMA buffer descriptor. The next table explains each buffer descriptor component.

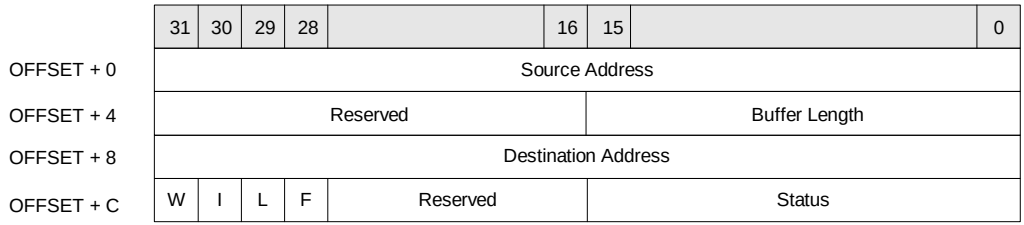


Figure 87: DMA buffer descriptor

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.
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 actual receive data byte count. **Note:** The receive 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 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. For other types of transfers, this field identifies the starting location of the destination data buffer. The destination address can start on any byte boundary.
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 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 319: 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), software must always set this field to 1. For *USB-OUT* transactions (DMA write), software must never set this field 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 must also 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.

Table 319: DMA buffer descriptor definition

DMA transfer status

The DMA buffer descriptor status field is updated when the buffer descriptor is retired. Tables 320 through 325 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 320: 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
11:01	UNUSED	Not used — read back 0
00	ROVER	Receive overrun error

Table 321: Peripheral bit fields: Serial controller — SPI RX mode

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 322: Peripheral bit fields: Serial controller — UART TX mode

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 323: Peripheral bit fields: Serial controller — SPI TX mode

Bits	Mnemonic	Description
15:14	UNUSED	Not used — read back 0
13	M31	See the USB Device Module chapter.
12	M30	See the USB Device Module chapter.
11	OVFLOW	Valid only for *out* direction endpoints. This bit is set if the endpoints' FIFO reaches a full state at any point during a transfer. This field reads back 0 for *in* direction endpoints.
10	TIMEOUT	Valid only for *out* direction endpoints. This bit is set if the buffer was closed due to a timeout event. This field reads back 0 for *in* direction endpoints.
09:00	UNUSED	Not used — read back as 0

Table 324: Peripheral bit fields: USB controller

Bits	Mnemonic	Description
15:00	UNUSED	Not used — read back 0

Table 325: Peripheral bit fields: IEEE 1284 controller

DMA channel assignments

Each of the two BBus DMA controllers 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.

The next table 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 A transmitter	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 CMD receiver	FBW
DMA1	10	1284 CMD transmitter	FBR
DMA1	11	1284 data receiver	FBW
DMA1	12	1284 data transmitter	FBR
DMA1	13	Not used	N/A
DMA1	14	Not used	N/A
DMA1	15	Not used	N/A
DMA1	16	Not used	N/A
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

Table 326: DMA channel assignments

DMA	Channel	DMA channel peripheral	Fly-by direction
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	Not used	N/A
DMA2	14	Not used	N/A
DMA2	15	Not used	N/A
DMA2	16	Not used	N/A

Table 326: 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 0x9091 0000.

Table 327 is a single DMA controller address map.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

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 / 9091 0000	DMA Channel 1 Buffer Descriptor Pointer
9000 0020 / 9091 0020	DMA Channel 2 Buffer Descriptor Pointer
9000 0040 / 9091 0040	DMA Channel 3 Buffer Descriptor Pointer
9000 0060 / 9091 0060	DMA Channel 4 Buffer Descriptor Pointer
9000 0080 / 9091 0080	DMA Channel 5 Buffer Descriptor Pointer
9000 00A0 / 9091 00A0	DMA Channel 6 Buffer Descriptor Pointer
9000 00C0 / 9091 00C0	DMA Channel 7 Buffer Descriptor Pointer
9000 00E0 / 9091 00E0	DMA Channel 8 Buffer Descriptor Pointer
9000 0100 / 9091 0100	DMA Channel 9 Buffer Descriptor Pointer
9000 0120 / 9091 0120	DMA Channel 10 Buffer Descriptor Pointer
9000 0140 / 9091 0140	DMA Channel 11 Buffer Descriptor Pointer
9000 0160 / 9091 0160	DMA Channel 12 Buffer Descriptor Pointer
9000 0180 / 9091 0180	DMA Channel 13 Buffer Descriptor Pointer
9000 01A0 / 9091 01A0	DMA Channel 14 Buffer Descriptor Pointer
9000 01C0 / 9091 01C0	DMA Channel 15 Buffer Descriptor Pointer
9000 01E0 / 9091 01E0	DMA Channel 16 Buffer Descriptor Pointer
9000 0010 / 9091 0010	DMA Channel 1 Control register
9000 0030 / 9091 0030	DMA Channel 2 Control register
9000 0050 / 9091 0050	DMA Channel 3 Control register
9000 0070 / 9091 0070	DMA Channel 4 Control register
9000 0090 / 9091 0090	DMA Channel 5 Control register
9000 00B0 / 9091 00B0	DMA Channel 6 Control register
9000 00D0 / 9091 00D0	DMA Channel 7 Control register
9000 00F0 / 9091 00F0	DMA Channel 8 Control register
9000 0110 / 9091 0110	DMA Channel 9 Control register
9000 0130 / 9091 0130	DMA Channel 10 Control register

Table 327: DMA Control and Status register address map

Offset	Description
9000 0150 / 9091 0150	DMA Channel 11 Control register
9000 0170 / 9091 0170	DMA Channel 12 Control register
9000 0190 / 9091 0190	DMA Channel 13 Control register
9000 01B0 / 9091 01B0	DMA Channel 14 Control register
9000 01D0 / 9091 01D0	DMA Channel 15 Control register
9000 01F0 / 9091 01F0	DMA Channel 16 Control register
9000 0014 / 9091 0014	DMA Channel 1 Status/Interrupt Enable register
9000 0034 / 9091 0034	DMA Channel 2 Status/Interrupt Enable register
9000 0054 / 9091 0054	DMA Channel 3 Status/Interrupt Enable register
9000 0074 / 9091 0074	DMA Channel 4 Status/Interrupt Enable register
9000 0094 / 9091 0094	DMA Channel 5 Status/Interrupt Enable register
9000 00B4 / 9091 00B4	DMA Channel 6 Status/Interrupt Enable register
9000 00D4 / 9091 00D4	DMA Channel 7 Status/Interrupt Enable register
9000 00F4 / 9091 00F4	DMA Channel 8 Status/Interrupt Enable register
9000 0114 / 9091 0114	DMA Channel 9 Status/Interrupt Enable register
9000 0134 / 9091 0134	DMA Channel 10 Status/Interrupt Enable register
9000 0154 / 9091 0154	DMA Channel 11 Status/Interrupt Enable register
9000 0174 / 9091 0174	DMA Channel 12 Status/Interrupt Enable register
9000 0194 / 9091 0194	DMA Channel 13 Status/Interrupt Enable register
9000 01B4 / 9091 01B4	DMA Channel 14 Status/Interrupt Enable register
9000 01D4 / 9091 01D4	DMA Channel 15 Status/Interrupt Enable register
9000 01F4 / 9091 01F4	DMA Channel 16 Status/Interrupt Enable register

Table 327: DMA Control and Status register address map

DMA Buffer Descriptor Pointer

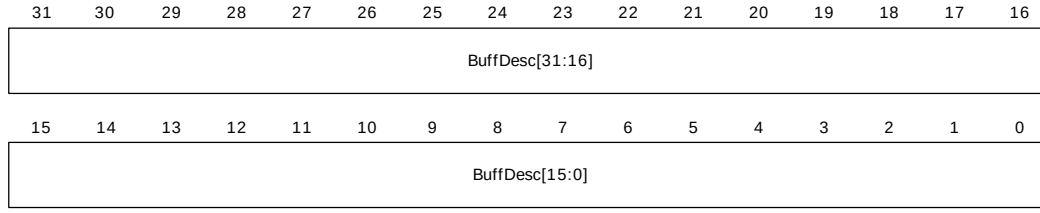
Address: DMA1

9000 0000 / 0020 / 0040 / 0060 / 0080 / 00A0 / 00C0 / 00E0 / 0100 / 0120 / 0140 / 0160 / 0180 / 01A0 / 01C0 / 01E0

Address: DMA2

9091 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 328: BBus DMA Buffer Descriptor Pointer register bit definition

DMA Control register

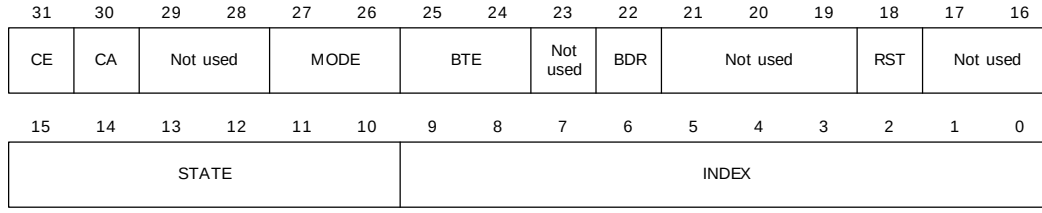
Address: DMA1

9000 0010 / 0030 / 0050 / 0070 / 0090 / 00B0 / 00D0 / 00F0 / 0110 / 0130 / 0150 / 0170 / 0190 / 01B0 / 01D0 / 01F0

Address: DMA2

9091 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.



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	Not used	0	Always set to 0.
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.

Table 329: BBus DMA Control register bit definition

Bits	Access	Mnemonic	Reset	Description
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	Not used	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.
D21:19	R/W	Not used	0	Always set to 0.
D18	R/W	RST	0	**Reset** Forces a reset of the DMA channel. Writing a 1 to this field forces all fields in this register, *except the index field*, to the reset state. The index field is written with the value specified on signals bbus_data[9:0]. Writing a 1 to this field while the DMA channel is operational can have unpredictable results.
D17:16	R/W	Not used	0	Always set to 0.
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/W	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 329: BBus DMA Control register bit definition

DMA Status/Interrupt Enable register

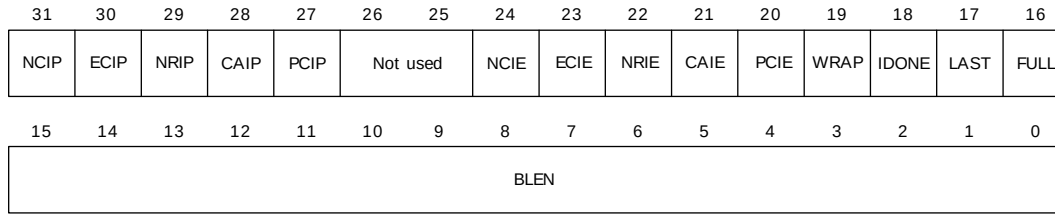
Address: DMA1

9000 0014 / 0034 / 0054 / 0074 / 0094 / 00B4 / 00D4 / 00F4 / 0114 / 0134 / 0154 / 0174 / 0194 / 01B4 / 01D4 / 01F4

Address: DMA2

9091 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.

Table 330: DMA Status/Interrupt Enable register bit definition

Bits	Access	Mnemonic	Reset	Description
D30	RW1TC	ECIP	0	**Error completion interrupt pending** Generated when one of these conditions occurs: ■ An invalid address is programmed in the Buffer Descriptor Pointer register. ■ An invalid receive buffer address is programmed in the source address field of a buffer descriptor. ■ The MODE field in the DMA Control register is set to an undefined value. Note that an ECIP is not generated when an invalid transmit buffer address is programmed in the source address field of a buffer descriptor. In this case, the AHB bus monitor signals the errored memory access. An interrupt will be generated if the ECIE (D23) bit is set. The DMA channel stops until firmware writes the CE bit (in the DMA Channel Control register) to a 1. 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. 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 330: 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	Not used	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 should always 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 should always 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 330: DMA Status/Interrupt Enable register bit definition

BBus Utility

CHAPTER 9

The BBus utility provides chip-level support for the low speed peripherals in the NS9360 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 NS9360 ASIC contains 73 pins that are designated as general purpose input/output (GPIO) pins. All signals driven by the BBus Utility module to a disabled peripheral are held in the inactive state.

The BBus Utility configuration registers are located at base address 0x9060 0000. Table 331 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.

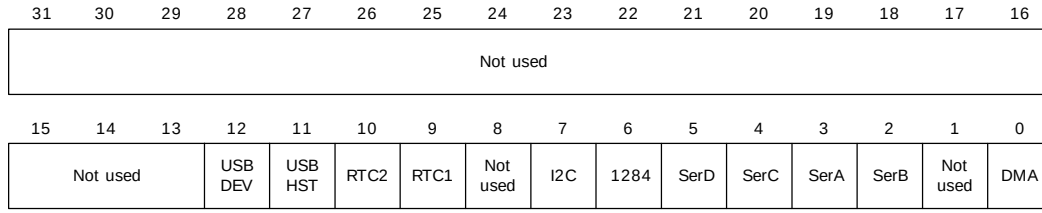
Address	Description
9060 0000	Master Reset register
9060 0004	BBus Utility Interrupt Status register
9060 0010	GPIO Configuration Register #1
9060 0014	GPIO Configuration Register #2
9060 0018	GPIO Configuration Register #3
9060 001C	GPIO Configuration Register #4
9060 0020	GPIO Configuration Register #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 Timeout 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
9060 0100	GPIO Configuration Register #8
9060 0104	GPIO Configuration Register #9
9060 0108	GPIO Configuration Register #10
9060 0120	GPIO Control Register #3
9060 0130	GPIO Status Register #3

Table 331: 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.



Register bit assignment

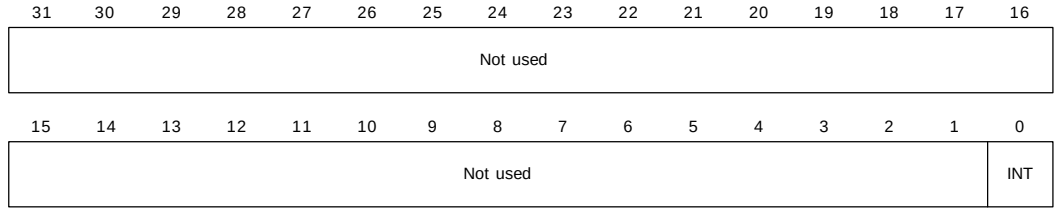
Bits	Access	Mnemonic	Reset	Description
D31:13	R/W	Not used	0	Always write to 0.
D12	R/W	USBDEV	1	USB Device controller reset
D11	R/W	USBHST	1	USB Host controller reset
D10	R/W	RTC2	1	RTC 12/24 hour calendar reset
D09	R/W	RTC1	1	RTC configuration registers reset
D08	R/W	Not used	1	Must write to 0.
D07	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	Not used	0	Always write to 1.
D00	R/W	DMA	1	BBus DMA reset

Table 332: Master Reset register

BBus Utility Interrupt Status register

Address: 9060 0004

The BBus Utility Status register contains the interrupt status information from the BBus peripherals. *All interrupts must be serviced in the originating module.*



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:01	R	Not used	0x0000_0000	Always read as 0x0000_0000.
D00	RW1TC	INT	0	A 1 read in this field indicates that the BBus Utility module has asserted its interrupt. The system must write a 1 to this bit to clear the interrupt.

Table 333: BBus Utility Interrupt Status register

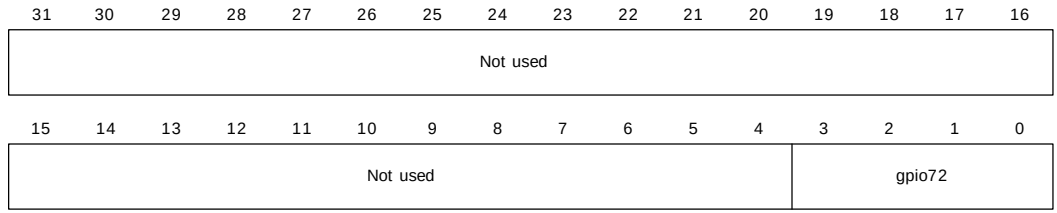
GPIO Configuration registers

GPIO Configuration registers #1 – #10 contain configuration information for each of the 73 GPIO pins in the NS9360. Each GPIO pin can be configured to serve up to four functions. Configure each pin for the appropriate function and direction, as shown in Table 344: "GPIO Configuration register options" on page 473.

Please refer to Table 1 in Chapter 1 for more information on reset behavior.

GPIO Configuration Register #10

Address: 9060 0108

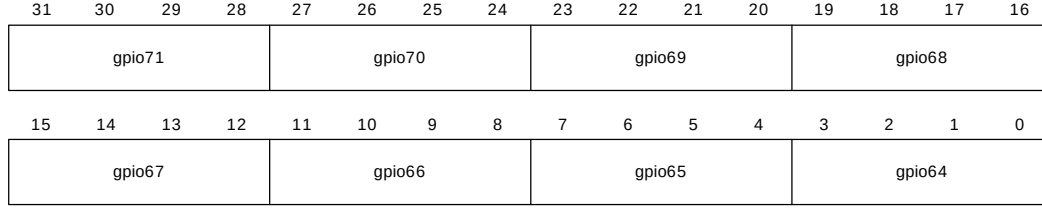


Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	Not used	0x3333333	Always write 0x3333333.
D03:00	R/W	gpio72	0x3	gpio[72] configuration

Table 334: GPIO Configuration Register #10

GPIO Configuration Register #9

Address: 9060 0104

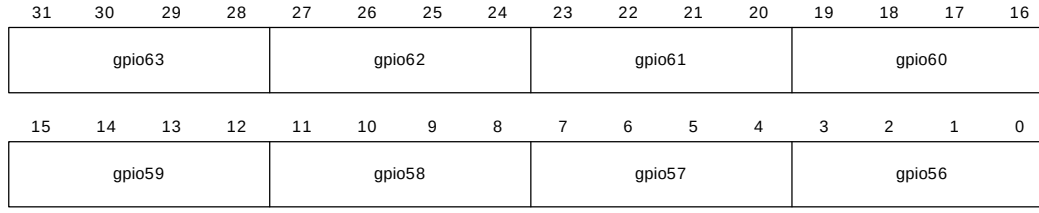


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio71	0x3	gpio[71] configuration
D27:24	R/W	gpio70	0x3	gpio[70] configuration
D23:20	R/W	gpio69	0x3	gpio[69] configuration
D19:16	R/W	gpio68	0x3	gpio[68] configuration
D15:12	R/W	gpio67	0x3	gpio[67] configuration
D11:08	R/W	gpio66	0x3	gpio[66] configuration
D07:04	R/W	gpio65	0x3	gpio[65] configuration
D03:00	R/W	gpio64	0x3	gpio[64] configuration

Table 335: GPIO Configuration Register #9

GPIO Configuration Register #8

Address: 9060 0100



Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio63	0x3	gpio[63] configuration
D27:24	R/W	gpio62	0x3	gpio[62] configuration
D23:20	R/W	gpio61	0x3	gpio[61] configuration
D19:16	R/W	gpio60	0x3	gpio[60] configuration
D15:12	R/W	gpio59	0x3	gpio[59] configuration
D11:08	R/W	gpio58	0x3	gpio[58] configuration

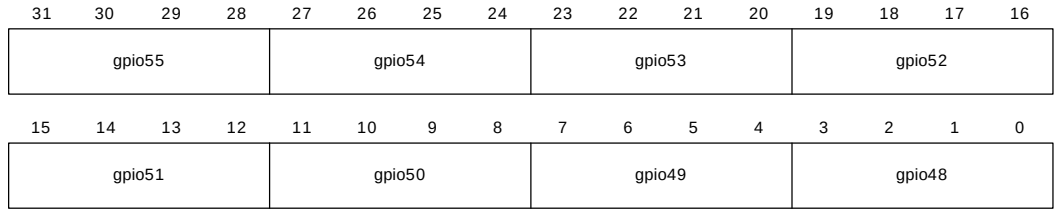
Table 336: GPIO Configuration Register #8

Bits	Access	Mnemonic	Reset	Description
D07:04	R/W	gpio57	0x3	gpio[57] configuration
D03:00	R/W	gpio56	0x3	gpio[56] configuration

Table 336: GPIO Configuration Register #8

GPIO Configuration Register #7

Address: 9060 0028

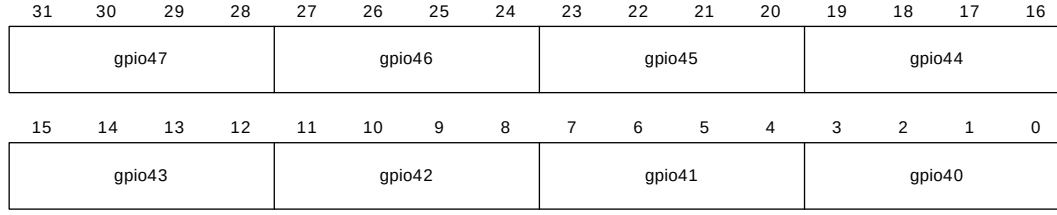


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio55	0x3	gpio[55] configuration
D27:24	R/W	gpio54	0x3	gpio[54] configuration
D23:20	R/W	gpio53	0x3	gpio[53] configuration
D19:16	R/W	gpio52	0x3	gpio[52] configuration
D15:12	R/W	gpio51	0x3	gpio[51] configuration
D11:08	R/W	gpio50	0x3	gpio[50] configuration
D07:04	R/W	gpio49	0x3	gpio[49] configuration
D03:00	R/W	gpio48	0x3	gpio[48] configuration

Table 337: GPIO Configuration Register #7

GPIO Configuration Register #6

Address: 9060 0024

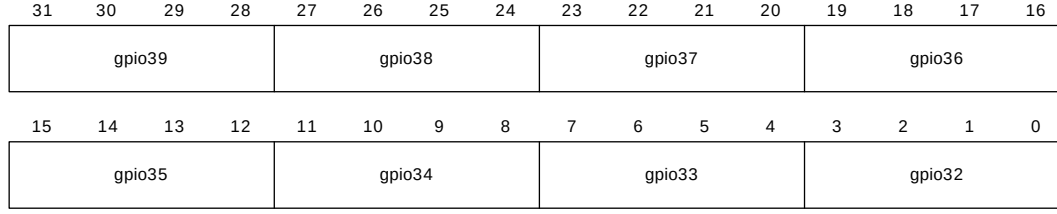


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio47	0x3	gpio[47] configuration
D27:24	R/W	gpio46	0x3	gpio[46] configuration
D23:20	R/W	gpio45	0x3	gpio[45] configuration
D19:16	R/W	gpio44	0x3	gpio[44] configuration
D15:12	R/W	gpio43	0x3	gpio[43] configuration
D11:08	R/W	gpio42	0x3	gpio[42] configuration
D07:04	R/W	gpio41	0x3	gpio[41] configuration
D03:00	R/W	gpio40	0x3	gpio[40] configuration

Table 338: 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

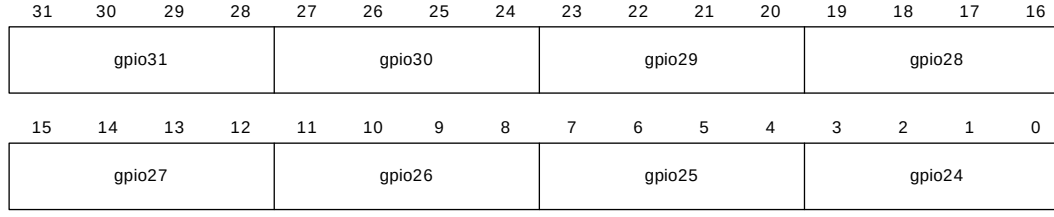
Table 339: GPIO Configuration Register #5

Bits	Access	Mnemonic	Reset	Description
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 339: GPIO Configuration Register #5

GPIO Configuration Register #4

Address: 9060 001C

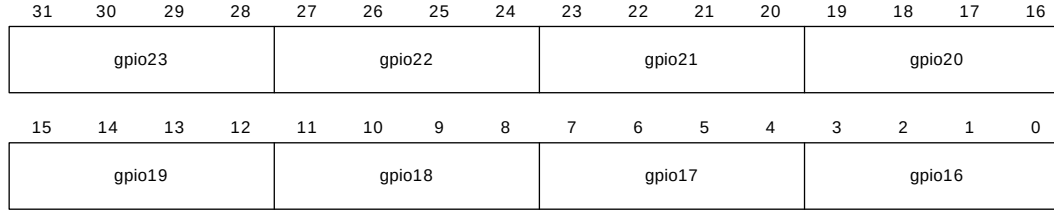


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio31	0x3	gpio[31] configuration
D27:24	R/W	gpio30	0x3	gpio[30] configuration
D23:20	R/W	gpio29	0x3	gpio[29] configuration
D19:16	R/W	gpio28	0x3	gpio[28] configuration
D15:12	R/W	gpio27	0x3	gpio[27] configuration
D11:08	R/W	gpio26	0x3	gpio[26] configuration
D07:04	R/W	gpio25	0x3	gpio[25] configuration
D03:00	R/W	gpio24	0x3	gpio[24] configuration

Table 340: GPIO Configuration Register #4

GPIO Configuration Register #3

Address: 9060 0018

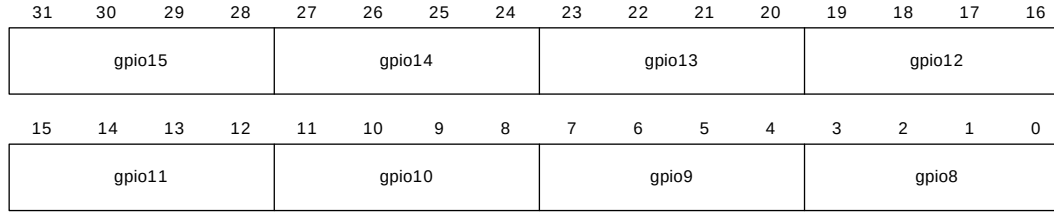


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio23	0x3	gpio[23] configuration
D27:24	R/W	gpio22	0x3	gpio[22] configuration
D23:20	R/W	gpio21	0x3	gpio[21] configuration
D19:16	R/W	gpio20	0x3	gpio[20] configuration
D15:12	R/W	gpio19	0x3	gpio[19] configuration
D11:08	R/W	gpio18	0x3	gpio[18] configuration
D07:04	R/W	gpio17	0x3	gpio[17] configuration
D03:00	R/W	gpio16	0x3	gpio[16] configuration

Table 341: GPIO Configuration Register #3

GPIO Configuration Register #2

Address: 9060 0014

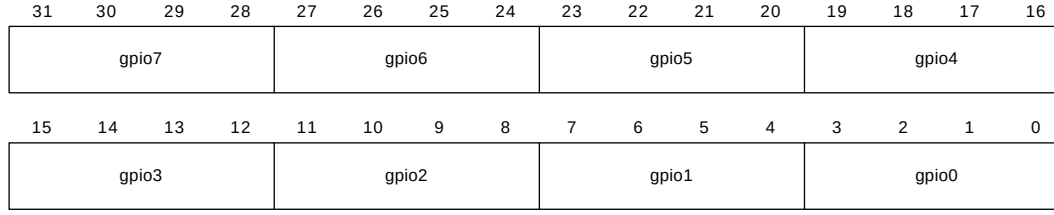


Bits	Access	Mnemonic	Reset	Description
D31:28	R/W	gpio15	0x3	gpio[15] configuration
D27:24	R/W	gpio14	0x3	gpio[14] configuration
D23:20	R/W	gpio13	0x3	gpio[13] configuration
D19:16	R/W	gpio12	0x3	gpio[12] configuration
D15:12	R/W	gpio11	0x3	gpio[11] configuration
D11:08	R/W	gpio10	0x3	gpio[10] configuration
D07:04	R/W	gpio9	0x3	gpio[9] configuration
D03:00	R/W	gpio8	0x3	gpio[8] configuration

Table 342: GPIO Configuration Register #2

GPIO Configuration Register #1

Address: 9060 0010



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 343: GPIO Configuration Register #1

GPIO Configuration register options

Bits	Mnemonic	Description
D03	DIR	Controls the pin direction when the FUNC field (D01:00) is configured for GPIO mode, function 3. 0 Input 1 Output The pin direction is controlled by function 0, 1, or 2, as selected in the FUNC field. ALL GPIO are reset to the input state.
D02	INV	Controls the inversion function of the GPIO pin. 0 Disables the inversion function 1 Enables the inversion function This bit applies to any of the functional modes
D01:00	FUNC	Use these bits to select the function. See the discussion of GPIO MUX, in the Pinout chapter, for details about the available pin functions. 00 Function #0 01 Function #1 10 Function #2 11 Function #3

Table 344: GPIO Configuration register options

GPIO Control registers

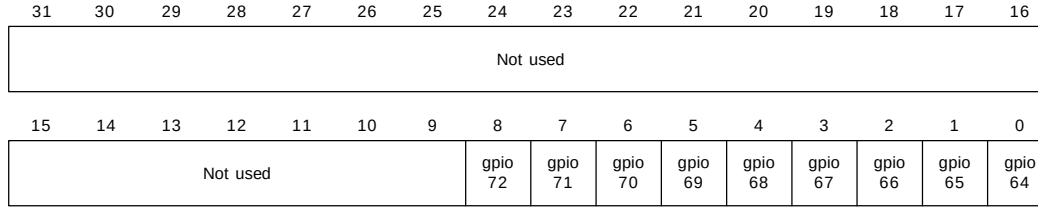
GPIO Control Registers #1 through #3 contain the control information for each of the 73 GPIO pins in the NS9360.

When a GPIO pin is configured as a GPIO output, the corresponding bit in GPIO Control Registers #1, #2, and #3 is driven out the GPIO pin. In all configurations, the CPU has read/write access to the register.

Please refer to Table 1 in Chapter 1 for more information on reset behavior.

GPIO Control Register #3

Address: 9060 0120

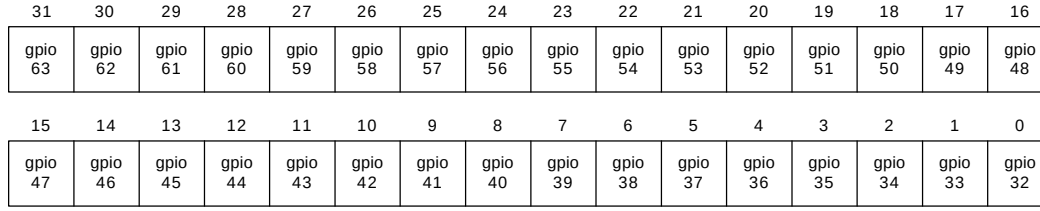


Bits	Access	Mnemonic	Reset	Description
D31:09	R/W	Not used	0	Always write to 0.
D08	R/W	gpio72	0	gpio[72] control bit
D07	R/W	gpio71	0	gpio[71] control bit
D06	R/W	gpio70	0	gpio[70] control bit
D05	R/W	gpio69	0	gpio[69] control bit
D04	R/W	gpio68	0	gpio[68] control bit
D03	R/W	gpio67	0	gpio[67] control bit
D02	R/W	gpio66	0	gpio[66] control bit
D01	R/W	gpio65	0	gpio[65] control bit
D00	R/W	gpio64	0	gpio[64] control bit

Table 345: GPIO Control Register #3

GPIO Control Register#2

Address: 9060 0034



Bits	Access	Mnemonic	Reset	Description
D31	R/W	gpio63	0	gpio[63] control bit
D30	R/W	gpio62	0	gpio[62] control bit
D29	R/W	gpio61	0	gpio[61] control bit
D28	R/W	gpio60	0	gpio[60] control bit
D27	R/W	gpio59	0	gpio[59] control bit
D26	R/W	gpio58	0	gpio[58] control bit
D25	R/W	gpio57	0	gpio[57] control bit
D24	R/W	gpio56	0	gpio[56] control bit
D23	R/W	gpio55	0	gpio[55] control bit
D22	R/W	gpio54	0	gpio[54] control bit
D21	R/W	gpio53	0	gpio[53] control bit
D20	R/W	gpio52	0	gpio[52] control bit
D19	R/W	gpio51	0	gpio[51] control bit
D18	R/W	gpio50	0	gpio[50] control bit
D17	R/W	gpio49	0	gpio[49] control bit
D16	R/W	gpio48	0	gpio[48] control bit
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

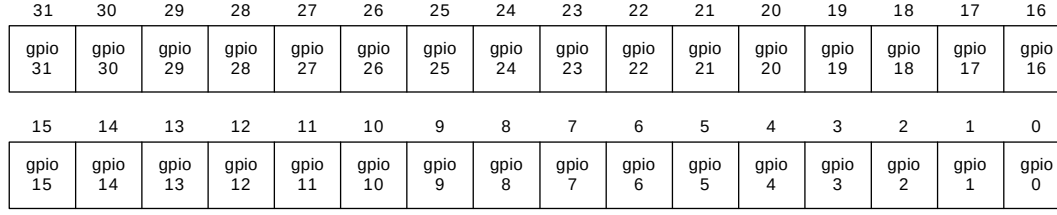
Table 346: GPIO Control Register #2

Bits	Access	Mnemonic	Reset	Description
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 346: GPIO Control Register #2

GPIO Control Register #1

Address: 9060 0030



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

Table 347: GPIO Control Register #1

Bits	Access	Mnemonic	Reset	Description
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
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 347: GPIO Control Register #1

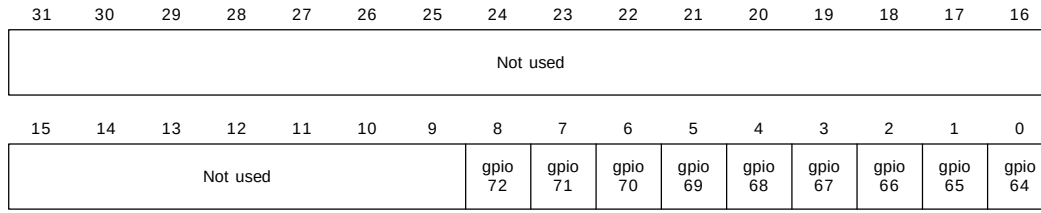
GPIO Status registers

GPIO Status Registers contain the status information for each of the 73 GPIO pins in the NS9360 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.

Note: The reset values for all of the status bits are undefined because they depend on the state of the GPIO pins to NS9360.

GPIO Status Register #3

Address: 9060 0130

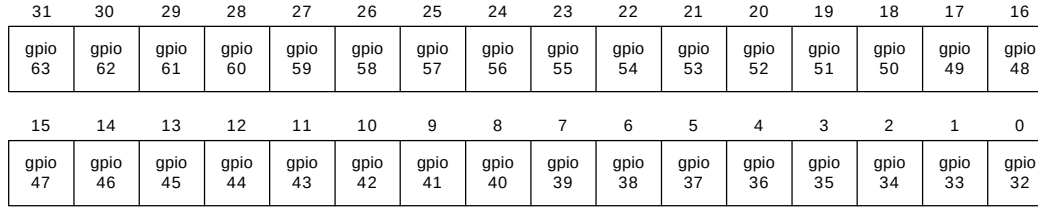


Bits	Access	Mnemonic	Reset	Description
D31:09	R	Not used	0	Always read as 0.
D08	R	gpio72	undefined	gpio[72] status bit
D07	R	gpio71	undefined	gpio[71] status bit
D06	R	gpio70	undefined	gpio[70] status bit
D05	R	gpio69	undefined	gpio[69] status bit
D04	R	gpio68	undefined	gpio[68] status bit
D03	R	gpio67	undefined	gpio[67] status bit
D02	R	gpio66	undefined	gpio[66] status bit
D01	R	gpio65	undefined	gpio[65] status bit
D00	R	gpio64	undefined	gpio[64] status bit

Table 348: GPIO Status Register #3

GPIO Status Register #2

Address: 9060 0044



Bits	Access	Mnemonic	Reset	Description
D31	R	gpio63	undefined	gpio[63] status bit
D30	R	gpio62	undefined	gpio[62] status bit
D29	R	gpio61	undefined	gpio[61] status bit
D28	R	gpio60	undefined	gpio[60] status bit
D27	R	gpio59	undefined	gpio[59] status bit
D26	R	gpio58	undefined	gpio[58] status bit
D25	R	gpio57	undefined	gpio[57] status bit
D24	R	gpio56	undefined	gpio[56] status bit
D23	R	gpio55	undefined	gpio[55] status bit
D22	R	gpio54	undefined	gpio[54] status bit
D21	R	gpio53	undefined	gpio[53] status bit
D20	R	gpio52	undefined	gpio[52] status bit
D19	R	gpio51	undefined	gpio[51] status bit
D18	R	gpio50	undefined	gpio[50] status bit
D17	R	gpio49	undefined	gpio[49] status bit
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

Table 349: GPIO Status Register #2

Bits	Access	Mnemonic	Reset	Description
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 349: 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

Bits	Access	Mnemonic	Reset	Description
D31	R	gpio31	undefined	gpio[31] status bit
D30	R	gpio30	undefined	gpio[30] status bit

Table 350: GPIO Status Register #1

Bits	Access	Mnemonic	Reset	Description
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
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

Table 350: GPIO Status Register #1

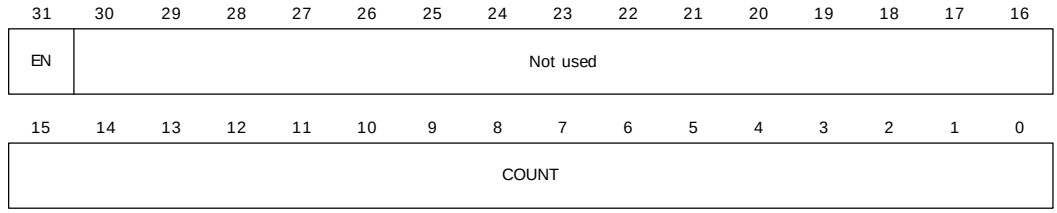
Bits	Access	Mnemonic	Reset	Description
D02	R	gpio2	undefined	gpio[2] status bit
D01	R	gpio1	undefined	gpio[1] status bit
D00	R	gpio0	undefined	gpio[0] status bit

Table 350: GPIO Status Register #1

BBus Timeout register

Address: 9060 0050

The BBus Timeout register contains the timeout information for the BBus activity monitor. The count field should always be set to at least 255.



Register bit assignment

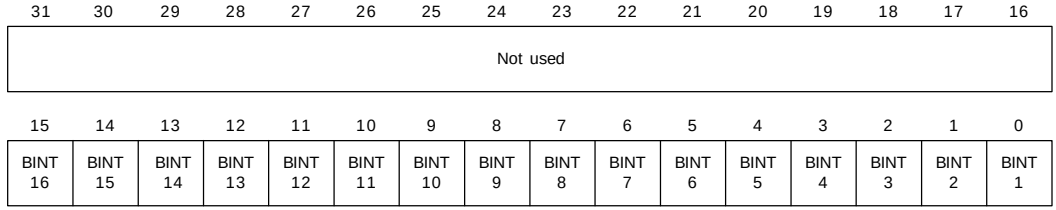
Bits	Access	Mnemonic	Reset	Description
D31	R/W	EN	1	Active *high BBus* monitor enable.
D30:16	R	Not used	0x0000	Always read as 0x0000
D15:00	R/W	COUNT	0x0FFF	The maximum number of cycles allotted to a BBus cycle.

Table 351: BBus TImeout 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.



Register bit assignment

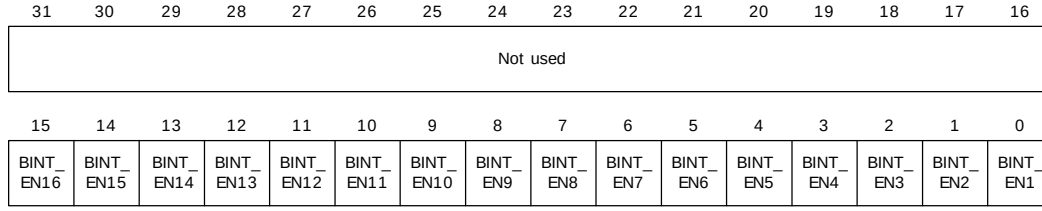
Bits	Access	Mnemonic	Reset	Description
D31:16	R	Not used	0x0000	Always read as 0x0000.
D15	R	BINT16	0	BBus DMA channel #16 interrupt status
D14	R	BINT15	0	BBus DMA channel #15 interrupt status
D13	R	BINT14	0	BBus DMA channel #14 interrupt status
D12	R	BINT13	0	BBus DMA channel #13 interrupt status
D11	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
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 352: 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R	Not used	0x0000	Always read as 0x0000
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
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

Table 353: BBus DMA Interrupt Enable register

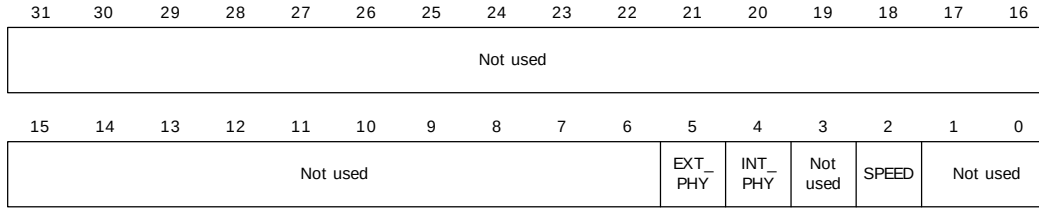
Bits	Access	Mnemonic	Reset	Description
D01	R/W	BINT_EN2	0	BBus DMA channel #2 interrupt enable
D00	R/W	BINT_EN1	0	BBus DMA channel #1 interrupt enable

Table 353: 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 464).



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:06	R	Not used	0	Always read as 0.
D05	R/W	EXT_PHY	0	Defines the type of data bus between the NS9360 and the external USB PHY. 0 The data bus is bidirectional. 1 The data bus is unidirectional.
D04	R/W	INT_PHY	0	Defines which USB module is currently using the on-chip PHY. 0 USB Host 1 USB Device
D03	R	Not used	0	Always read as 0.

Table 354: USB Configuration register

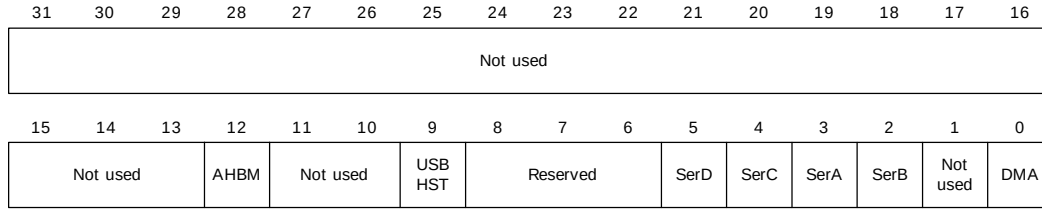
Bits	Access	Mnemonic	Reset	Description
D02	R/W	SPEED	1	Defines the operational speed of the USB Device block. 0 Low speed (1.5 Mbps) 1 Full speed (12 Mbps)
D01:00	R	Not used	0	Always read as 0.

Table 354: 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. NS9360 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:13	R	Not used	0	Always read as 0.
D12	R/W	AHBM	Reset to the value provided on strapping pin gpio[44]	**AHB bus master** 0 Little endian 1 Big endian

Table 355: Endian Configuration register

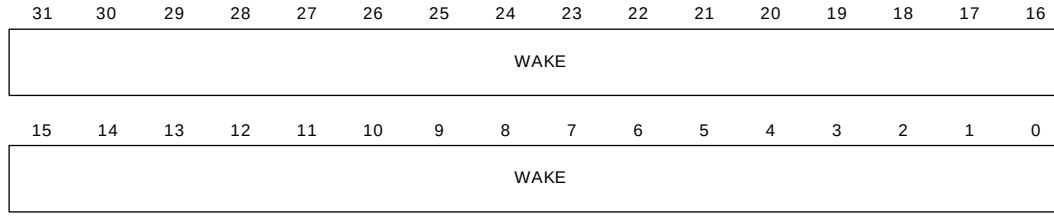
Bits	Access	Mnemonic	Reset	Description
D11:10	N/A	Reserved	N/A	N/A
D09	R/W	USBHST	Reset to the value provided on strapping pin gpio[44]	**USB Host controller** 0 Little endian 1 Big endian
D08:06	N/A	Reserved	N/A	N/A
D05	R/W	SerD	0	**Serial controller port D** 0 Little endian 1 Big endian
D04	R/W	SerC	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
D02	R/W	SerB	0	**Serial controller port B** 0 Little endian 1 Big endian
D01	R/W	Not used	0	Always write as 0.
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 355: 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	WAKE	0x0000_0000	Defines the word that must match in order for the Serial Controller to signal a wake-up to the ARM.

Table 356: ARM Wake-up register

Real Time Clock Module

CHAPTER 10

The Real Time Clock (RTC) module tracks the time of the day to an accuracy of 10 milliseconds and provides calendar functionality that tracks day, month, and year.

RTC functionality

RTC monitors these time periods:

- Year from 1900-2999
- Month from 1-12
- Date from 1-28, 29, 30, or 31, as a function of year and month
- Day of week from 1-7
- Hour from 0-23, or from 1-12 with the AM/PM flag set
- Minute from 0-59
- Second from 0.00-59.99

RTC functionality also provides an alarm register that allows comparison of month, date, hour, minute, second, and hundredth-second. Each item can be masked, allowing an alarm to be generated at a particular time and date on a monthly basis. An interrupt can be generated on the alarm event.

Event detection finds and generates interrupts on rollover conditions, including rollovers into a new month, date, hour, minute, second, or hundredth-second.

RTC configuration and status registers

Table 357 lists the RTC configuration and status registers.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

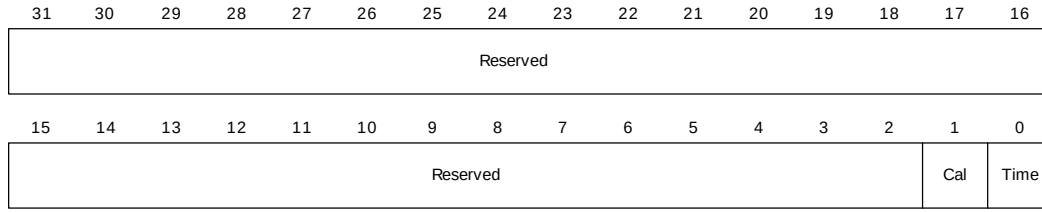
Address	Description
9070 0000	RTC General Control register
9070 0004	12/24 Hour register
9070 0008	Time register
9070 000C	Calendar register
9070 0010	Time Alarm register
9070 0014	Calendar Alarm register
9070 0018	Alarm Enable register
9070 001C	Event Flags register
9070 0020	Interrupt Enable register
9070 0024	Interrupt Disable register
9070 0028	Interrupt Status register
9070 002C	General Status register

Table 357: RTC configuration and status registers

RTC General Control register

Address: 9070 0000

The RTC General Control register contains miscellaneous settings for the RTC module.



Register bit assignment

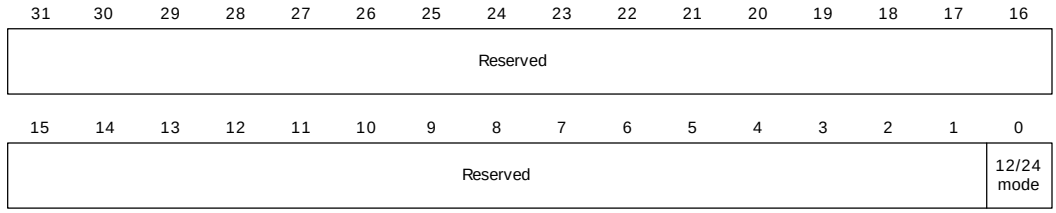
Bits	Access	Mnemonic	Reset	Description
D31:02	N/A	Reserved	N/A	N/A
D01	R/W	Cal	0x1	**Calendar operation** 0 Calendar operation enabled 1 Calendar operation disabled
D00	R/W	Time	0x1	**Time (date, hour, minute, second) operation** 0 Time operation enabled 1 Time operation disabled

Table 358: RTC General Control register

12/24 Hour register

Address: 9070 0004

The 12/24 Hour register controls 12 or 24 hour clock mode operation.



Register bit assignment

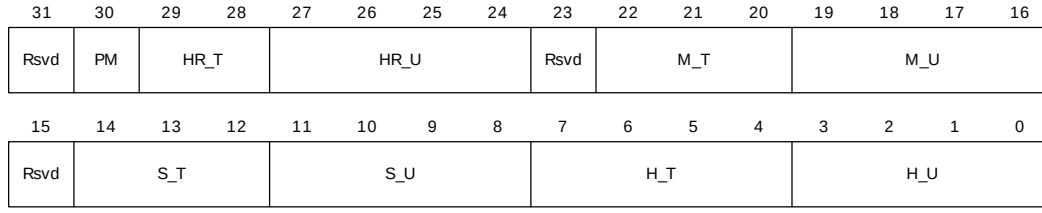
Bits	Access	Mnemonic	Reset	Description
D31:01	N/A	Reserved	N/A	N/A
D00	R/W	12/24	0x0	**12/24 clock mode operation** 0 24 hour mode operation 1 12 hour mode operation

Table 359: 12/24 Hour register

Time register

Address: 9070 0008

The TIme register sets the time values to the correct values, and reads the time registers. *BCD* is binary coded decimal.



Register bit assignment

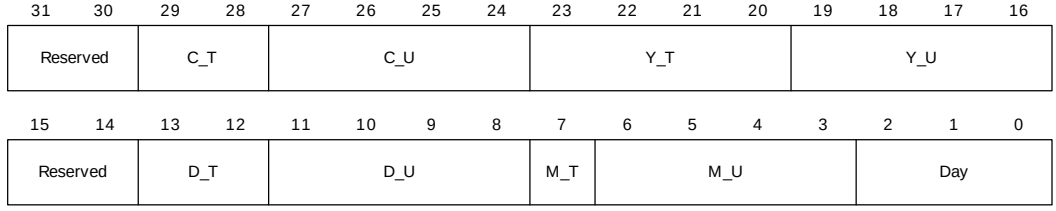
Bits	Access	Mnemonic	Reset	Description
D31	N/A	Reserved	N/A	N/A
D30	R/W	PM	0x0	**PM** Used in 12 hour mode only. 0 AM 1 PM
D29:28	R/W	HR_T	0x0	Hours, tens, BCD digit (0-2)
D27:24	R/W	HR_U	0x0	Hours, units, BCD digit (0-9)
D23	N/A	Reserved	N/A	N/A
D22:20	R/W	M_T	0x0	Minutes, tens, BCD digit (0-5)
D19:16	R/W	M_U	0x0	Minutes, units, BCD digit (0-9)
D15	N/A	Reserved	N/A	N/A
D14:12	R/W	S_T	0x0	Seconds, tens, BCD digit (0-5)
D11:08	R/W	S_U	0x0	Seconds, units, BCD digit (0-9)
D07:04	R/W	H_T	0x0	Hundredths of a second, tens, BCD digit (0-9)
D03:00	R/W	H_U	0x0	Hundredths of a second, units, BCD digit (0-9)

Table 360: Time register

Calendar register

Address: 9070 000C

The Calendar register sets the calendar values to the correct values, and reads the calendar registers. *BCD* is binary coded decimal.



Register bit assignment

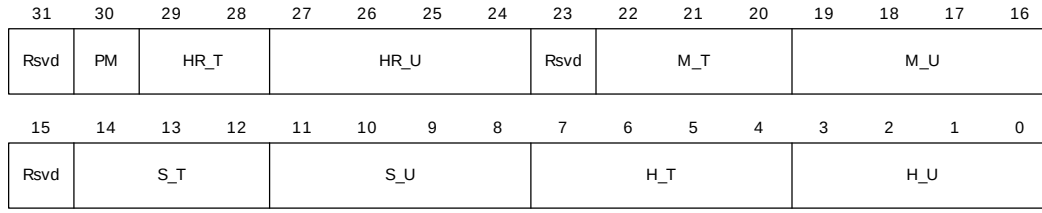
Bits	Access	Mnemonic	Reset	Description
D31:30	N/A	Reserved	N/A	N/A
D29:28	R/W	C_T	0x0	Century, tens, BCD digit (1-2)
D27:24	R/W	C_U	0x0	Century, units, BCD digit (0-9)
D23:20	R/W	Y_T	0x0	Years, tens, BCD digit (0-9)
D19:16	R/W	Y_U	0x0	Years, units, BCD digit (0-9)
D15:14	N/A	Reserved	N/A	N/A
D13:12	R/W	D_T	0x0	Date, tens, BCD digit (0-3)
D11:08	R/W	D_U	0x0	Date, units, BCD digit (0-9)
D07	R/W	M_T	0x0	Months, tens, BCD digit (0-1)
D06:03	R/W	M_U	0x0	Months, units, BCD digit (0-9)
D02:00	R/W	Day	0x0	Day of week, units, BCD digit (0-7)

Table 361: Calendar register

Time Alarm register

Address: 9070 0010

The Time Alarm register sets the time alarm. *BCD* is binary coded decimal.



Register bit assignment

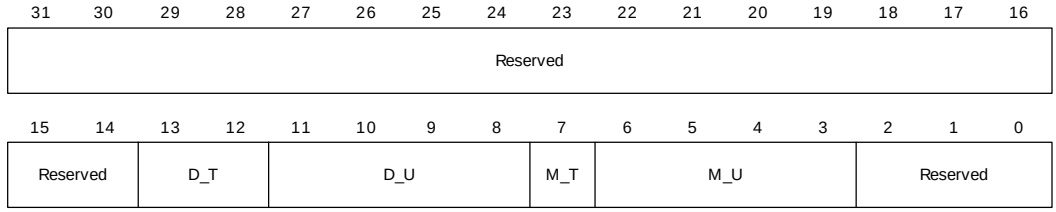
Bits	Access	Mnemonic	Reset	Description
D31	N/A	Reserved	N/A	N/A
D30	R/W	PM	0x0	**PM** Used in 12 hour mode only. 0 AM 1 PM
D29:28	R/W	HR_T	0x0	Hours, tens, BCD digit (0-2)
D27:24	R/W	HR_U	0x0	Hours, units, BCD digit (0-9)
D23	N/A	Reserved	N/A	N/A
D22:20	R/W	M_T	0x0	Minutes, tens, BCD digit (0-5)
D19:16	R/W	M_U	0x0	Minutes, units, BCD digit (0-9)
D15	N/A	Reserved	N/A	N/A
D14:12	R/W	S_T	0x0	Seconds, tens, BCD digit (0-5)
D11:08	R/W	S_U	0x0	Seconds, units, BCD digit (0-9)
D07:04	R/W	H_T	0x0	Hundredths of a second, tens, BCD digit (0-9)
D03:00	R/W	H_U	0x0	Hundredths of a second, units, BCD digit (0-9)

Table 362: Time Alarm register

Calendar Alarm register

Address: 9070 0014

The Calendar Alarm register sets the calendar alarm. This register programs a specific date and month when an alarm should cause an event. You cannot set an alarm that is more than one year in the future. *BCD* is binary coded decimal.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:14	N/A	Reserved	N/A	N/A
D13:12	R/W	D_T	0x0	Date, tens, BCD digit (0-3)
D11:08	R/W	D_U	0x0	Date, units, BCD digit (0-9)
D07	R/W	M_T	0x0	Months, tens, CD digit (0-1)
D06:03	R/W	M_U	0x0	Months, units, BCD digit (0-9)
D02:00	N/A	Reserved	N/A	N/A

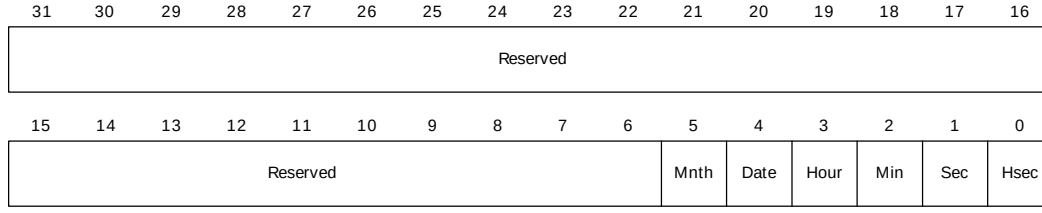
Table 363: Calendar Alarm register

Alarm Enable register

Address: 9070 0018

The Alarm Enable register sets the fields that can trigger an alarm. Setting a bit enables the corresponding time unit trigger event. Triggering the alarm causes an event to be generated, as set in the Events Flag register.

If all fields are enabled, an alarm is generated at the time set — the specific month, date, hour, minute, second, and hundredth-second. If only the minute field is set, the alarm triggers when that particular minute is reached, and every hour thereafter.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A
D05	R/W	Mnth	0x0	**Month** 0 Disable the month event 1 Enable the month event
D04	R/W	Date	0x0	**Date** 0 Disable the date event 1 Enable the date event
D03	R/W	Hour	0x0	**Hour** 0 Disable the hour event 1 Enable the hour event
D02	R/W	Min	0x0	**Minute** 0 Disable the minute event 1 Enable the minute event

Table 364: Alarm Enable register

Bits	Access	Mnemonic	Reset	Description
D01	R/W	Sec	0x0	**Second** 0 Disable the second event 1 Enable the second event
D00	R/W	Hsec	0x0	**Hundredth of a second** 0 Disable the hundredth second event 1 Enable the hundredth second event

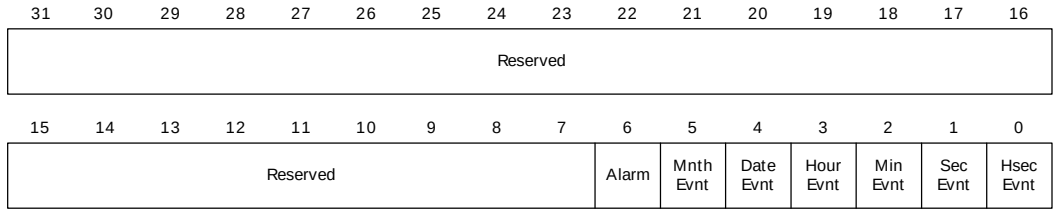
Table 364: Alarm Enable register

Event Flags register

Address: 9070 001C

The Event Flags register indicates that an event has occurred since the last reset. Read the register to determine the cause of the current active interrupt. This register is cleared when read (*R/R* in Access column).

Note that the Event Flags register can change even if the corresponding alarm enable bit is not set.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	R/R	Alarm	0x0	**Alarm event** One of the events programmed in the Alarm Events register has triggered.

Table 365: Event Flags register

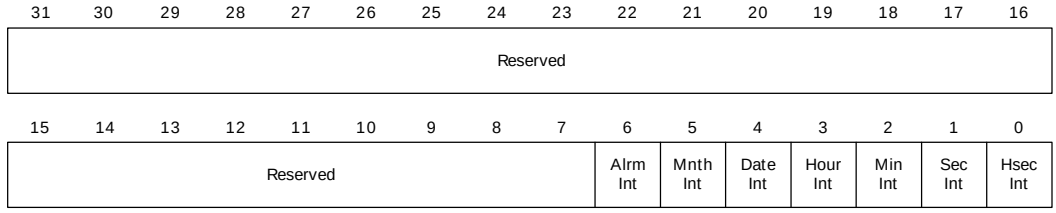
Bits	Access	Mnemonic	Reset	Description
D05	R/R	Mnth Evnt	0x0	**Month event** 0 Month event has not occurred 1 Month event has occurred
D04	R/R	Date Evnt	0x0	**Date event** 0 Date event has not occurred 1 Date event has occurred
D03	R/R	Hour Evnt	0x0	**Hour event** 0 Hour event has not occurred 1 Hour event has occurred
D02	R/R	Min Evnt	0x0	**Minute event** 0 Minute event has not occurred 1 Minute event has occurred
D01	R/R	Sec Evnt	0x0	**Second event** 0 Second event has not occurred 1 Second event has occurred
D00	R/R	Hsec Evnt	0x0	**Hundredth of a second event** 0 Hundredth second event has not occurred 1 Hundredth second event has occurred

Table 365: Event Flags register

Interrupt Enable register

Address: 9070 0020

The Interrupt Enable register sets which events can generate and interrupt. The interrupt that is generated remains set until it is cleared by disabling the event or by reading/clearing the Event Flags register.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	W	Alrm Int	0x0	**Alarm interrupt** 0 Disable alarm interrupt 1 Enable alarm interrupt
D05	W	Mnth Int	0x0	**Month interrupt** 0 Disable month interrupt 1 Enable month interrupt
D04	W	Date Int	0x0	**Date interrupt** 0 Disable date interrupt 1 Enable date interrupt
D03	W	Hour Int	0x0	**Hour interrupt** 0 Disable hour interrupt 1 Enable hour interrupt
D02	W	Min Int	0x0	**Minute interrupt** 0 Disable minute interrupt 1 Enable minute interrupt

Table 366: Interrupt Enable register

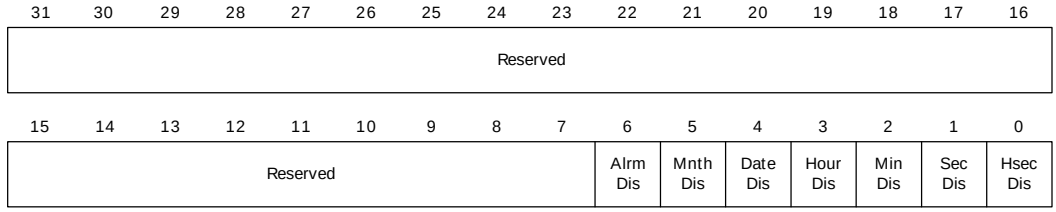
Bits	Access	Mnemonic	Reset	Description
D01	W	Sec Int	0x0	**Second interrupt** 0 Disable second interrupt 1 Enable second interrupt
D00	W	Hsec Int	0x0	**Hundredth of a second interrupt** 0 Disable hundredth second interrupt 1 Enable hundredth second interrupt

Table 366: Interrupt Enable register

Interrupt Disable register

Address: 9070 0024

The Interrupt Disable register resets interrupts that are currently enables. An interrupt is disabled by writing a 1, then a 0, to the appropriate disable register bit



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	W	Alrm Dis	0x0	**Alarm interrupt disable** 0 Enable alarm interrupt 1 Disable alarm interrupt
D05	W	Mnth Dis	0x0	**Month interrupt disable** 0 Enable month interrupt 1 Disable month interrupt
D04	W	Date Dis	0x0	**Date interrupt disable** 0 Enable date interrupt 1 Disable date interrupt
D03	W	Hour Dis	0x0	**Hour interrupt disable** 0 Enable hour interrupt 1 Disable hour interrupt
D02	W	Min Dis	0x0	**Minute interrupt disable** 0 Enable minute interrupt 1 Disable minute interrupt
D01	W	Sec Dis	0x0	**Second interrupt disable** 0 Enable second interrupt 1 Disable second interrupt

Table 367: Interrupt Disable register

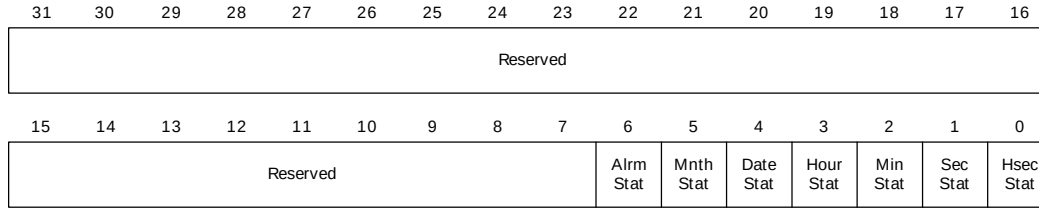
Bits	Access	Mnemonic	Reset	Description
D00	W	Hsec Dis	0x0	**Hundredth of a second interrupt disable** 0 Enable hundredth second interrupt 1 Disable hundredth second interrupt

Table 367: Interrupt Disable register

Interrupt Enable Status register

Address: 9070 0028

The Interrupt Enable Status register determines which interrupt sources are enabled and which interrupt sources are disabled.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	R	Alrm Stat	1	**Alarm interrupt status** 0 Interrupt enabled 1 Interrupt disabled
D05	R	Mnth Stat	1	**Month interrupt status** 0 Interrupt enabled 1 Interrupt disabled
D04	R	Date Stat	1	**Date interrupt status** 0 Interrupt enabled 1 Interrupt disabled

Table 368: Interrupt Enable Status register

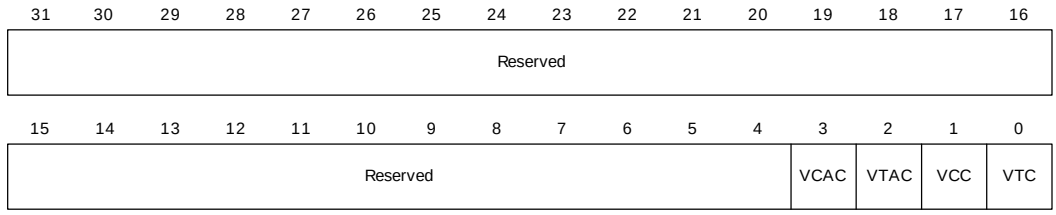
Bits	Access	Mnemonic	Reset	Description
D03	R	Hour Stat	1	**Hour interrupt status** 0 Interrupt enabled 1 Interrupt disabled
D02	R	Min Stat	1	**Minute interrupt status** 0 Interrupt enabled 1 Interrupt disabled
D01	R	Sec Stat	1	**Second interrupt status** 0 Interrupt enabled 1 Interrupt disabled
D00	R	Hsec Stat	1	**Hundredth of a second interrupt status** 0 Interrupt enabled 1 Interrupt disabled

Table 368: Interrupt Enable Status register

General Status register

Address: 9070 002C

The General Status register determines the status of the RTC configuration. If an invalid configuration is found, the RTC counters do not start operation.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A

Table 369: General Status register

Bits	Access	Mnemonic	Reset	Description
D03	R	VCAC	0x1	**Valid calendar alarm configuration** 0 Invalid 1 Valid
D02	R	VTAC	0x1	**Valid time alarm configuration** 0 Invalid 1 Valid
D01	R	VCC		**Valid calendar configuration** 0 Invalid 1 Valid
D00	R	VTC		**Valid time configuration** 0 Invalid 1 Valid

Table 369: General Status 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^2C module is designed to be a master and slave. The slave is active *only* when the module is being addressed during an I^2C bus transfer; the master can arbitrate for and access the I^2C bus *only* when the bus is free (idle) — therefore, the master and slave are mutually exclusive.

Physical I^2C bus

The physical I^2C bus consists of two open-drain signal lines: serial data (SDA) and serial clock (SCL). Pullup resistors are required; see the standard I^2C 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^2C bus.
- In receive mode (slave is written to), the I^2C bus interface receives 8-bit-based serial data from the I^2C 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^2C 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^2C 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.

I²C external addresses

I²C external [bus] addresses are allocated as two groups of eight addresses (0000XXX and 1111XXX):

Slave address	R/W bit	Description
0000 000	0	General call address
0000 000	1	START byte (not supported in NS9360)
0000 001	X	CBUS address (not supported in NS9360)
0000 010	X	Reserved for different bus format
0000 011	X	Reserved
0000 1xx	X	hs-mode master code (not supported in NS9360)
1111 1xx	X	Reserved
1111 0xx	X	10-bit slave address

Table 370: Reserved slave addresses

The general call address is for addressing all devices connected to the I²C 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.

I²C command interface

The I²C module converts parallel (8-bit) data to serial data and serial data to parallel data between the NS9360 and the I²C 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

I²C 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 521).

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) is sent. The master module can be sent a master command only; the slave module can be sent a slave command only (see "Master module and slave module commands," beginning on page 510, 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 I²C master recognizes four high-level commands, which are used in the CMD field of the Command register; the I²C slave recognizes two high-level commands:

Command	Name	Description
00_{hex}	M_NOP	No operation.

Table 371: Master and slave module commands

Command	Name	Description
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^2C 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 371: Master and slave module commands

Bus arbitration

Any M_READ or M_WRITE command causes the I^2C module to participate in the bus arbitration process when the I^2C 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.

I²C 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.

This table shows the register addresses. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

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

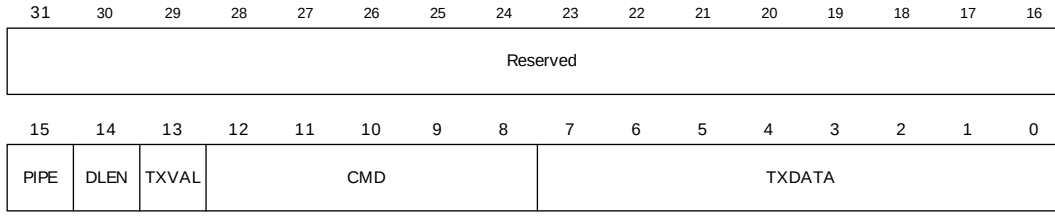
Table 372: I²C register address map

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 NS9360 BBus and I^2C bus. *This register is write only.*



Register bit assignment

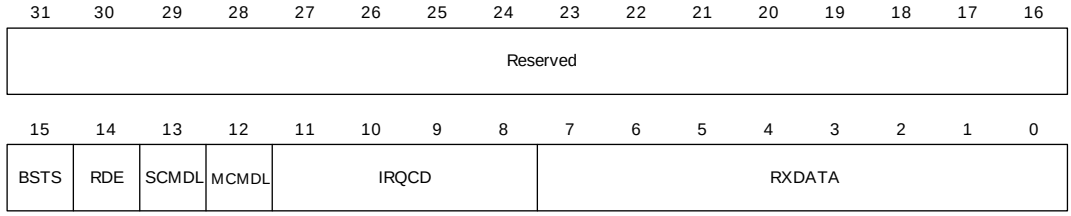
Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	**N/A**
D15	W	PIPE	00_{hex}	**Pipeline mode** Must be set to 0.
D14	W	DLEN	00_{hex}	**I^2C DLEN port** (iic_dlen) Must be set to 0.
D13	W	TXVAL	00_{hex}	**Provide new transmit data in** CMD_TX_DATA_REG (tx_data_val).
D12:08	W	CMD	00_{hex}	Command to be sent (see "Master module and slave module commands," beginning on page 510)
D07:00	W	TXDATA	00_{hex}	Transmit data to I^2C bus.

Table 373: 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 NS9360 BBus and I^2C bus. *This register is read only.*



Register bit assignment

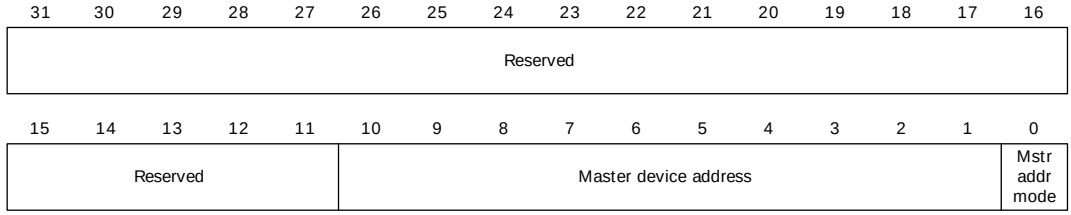
Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	**N/A**
D15	R	BSTS	N/A	**Bus status (master only)** 0 Bus is free 1 Bus is occupied
D14	R	RDE	N/A	**Receive data enable (**rx_data_en**)** Received data is available.
D13	R	SCMDL	N/A	**Slave command lock** The Slave Command register is locked.
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 518 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 378: "Master/slave interrupt codes" on page 518).

Table 374: 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.



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 375: 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.

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 376: 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:

$I^2C\_bus\_clock = clk / ((CLREF*2) + 4 + scl\_delay)$

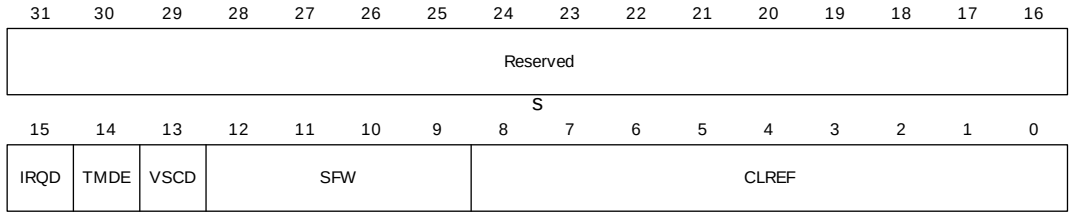
$clk = cpu\_clk/4$

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.

The timing parameter for fast-mode is as follows:

$I^2C\_bus\_clock = (4 / 3) \times (clk / ((CLREF*2) + 4 + scl\_delay))$

scl_delay is influenced by the SCL rise time.



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 377: 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 377: Configuration register

Interrupt Codes

Interrupts are signaled in the irq_code field in the STATUS_REG, by providing the appropriate interrupt code (see Table 378: "Master/slave interrupt codes" on page 518). 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 514).

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 378: 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 378: Master/slave interrupt codes

Software driver

The I^2C master software driver uses three commands only:

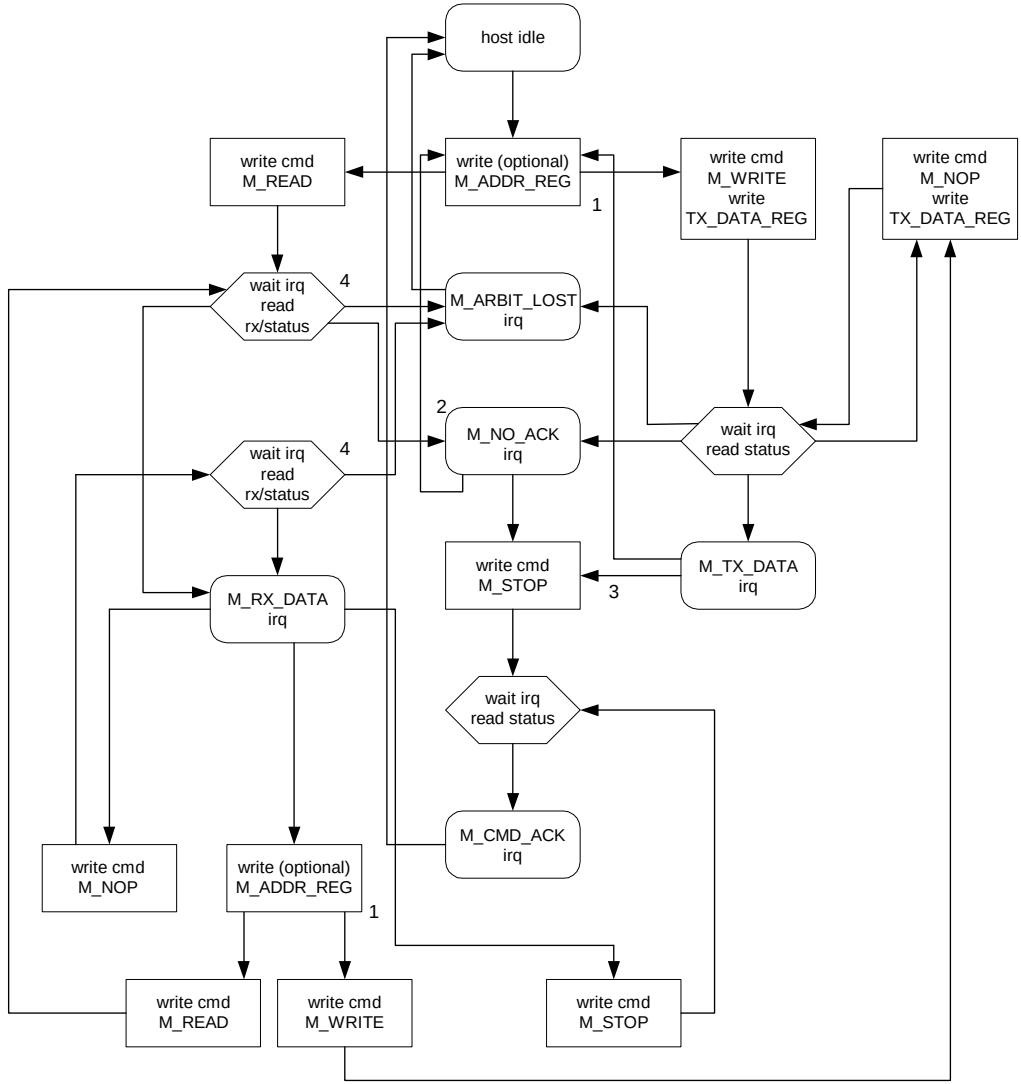
- M_READ to start a read sequence
- M_WRITE to start a write sequence
- M_STOP to give up the I^2C bus

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^2C bus. This opens the opportunity to provide a new slave device address in the MAster Address register before the command request.

The I^2C 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^2C 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 518). 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:

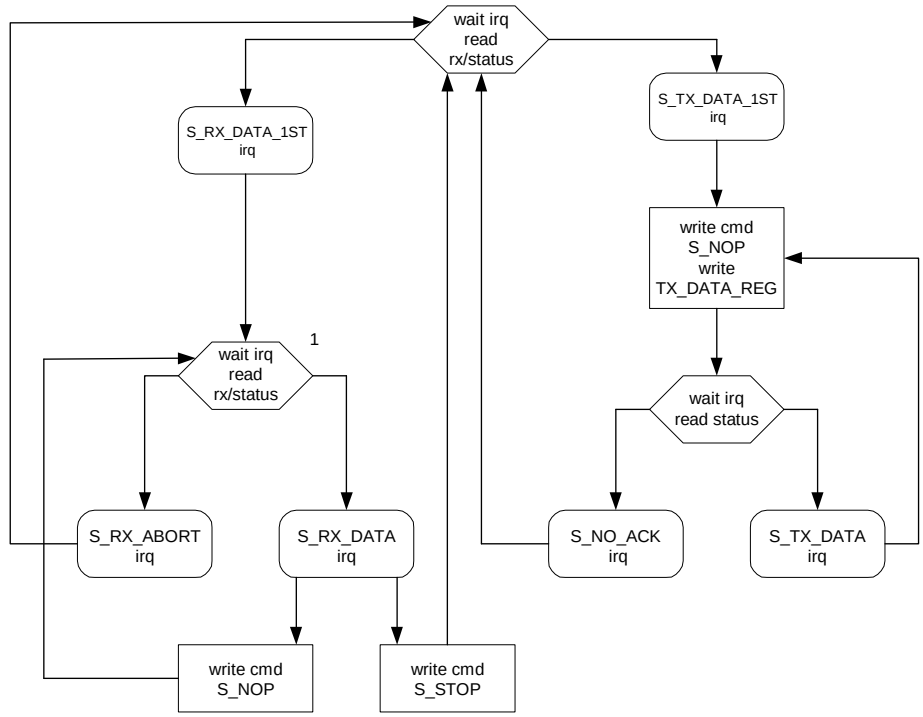
1 Writing M_ADDR_REQ is not required if the device address is not changed.

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

CHAPTER 12

The NS9360 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 are in the Timing chapter.

LCD features

The NS9360 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
- 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

Be sure that the maximum panel clock rate is in accord with your display panel's refresh requirement.

LCD panel support

The LCD controller supports these types of panels:

- Active matrix TFT panels with 18-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.

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 88, "Power up and power down sequences," on page 528):

1. V_{DD} is applied simultaneously to the NS9360 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[17:0], 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[17:0] 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[17:0] 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.

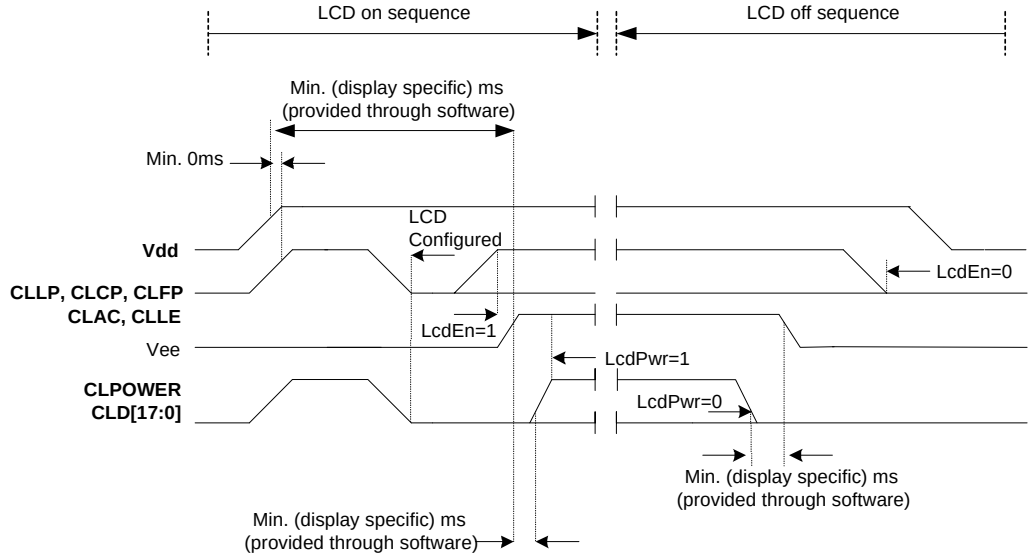


Figure 88: Power up and power down sequences

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 NS9360 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 379 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 NS9360. A *divided-by-2* version of this value is sent to the LCD controller.

Table 379: 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. 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.

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.

This figure shows the LCD controller module.

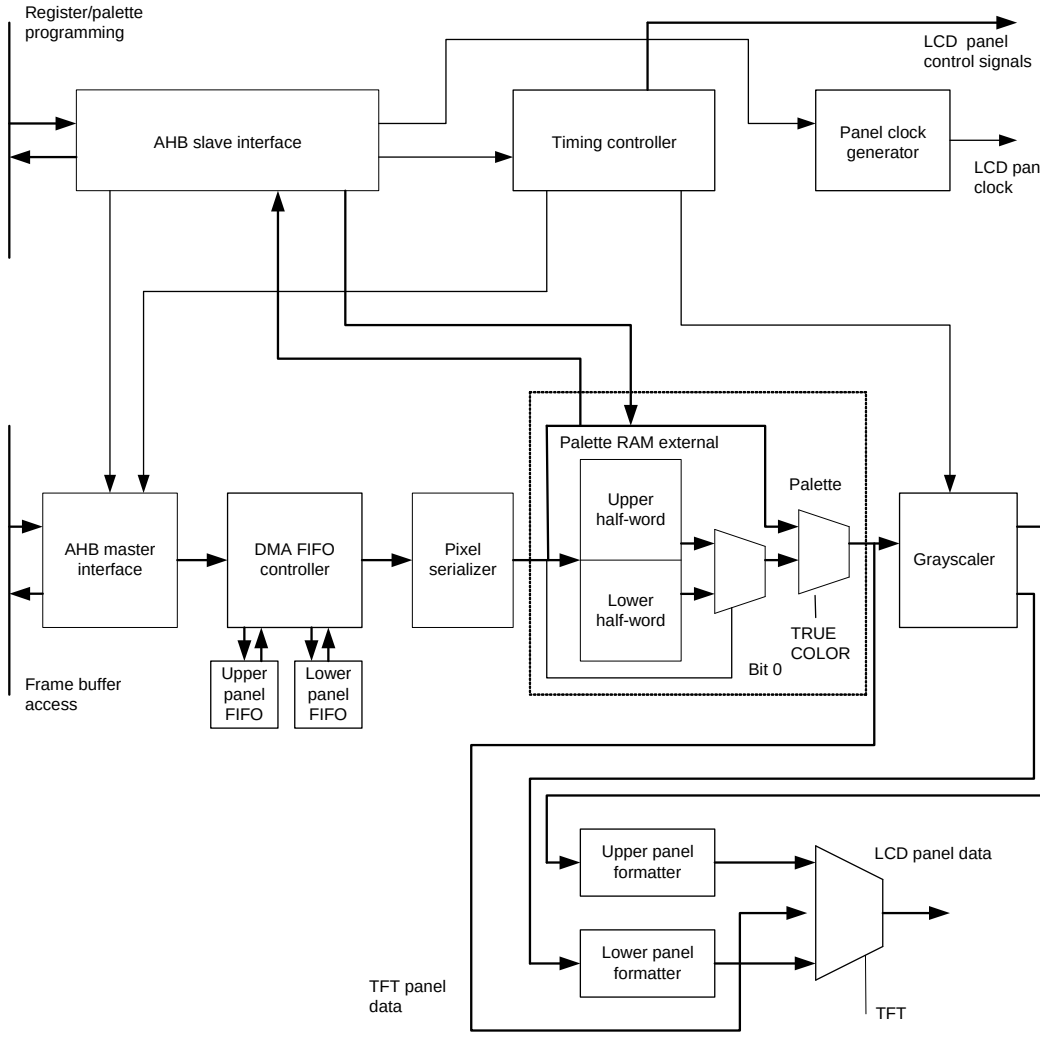


Figure 89: LCD controller block diagram

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 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 90 and Figure 91 show the data structure for little endian byte, little endian pixel — LBLP.

Figure 92 and Figure 93 show the data structure for big endian byte, big endian pixel — BBBP.

Figure 94 and Figure 95 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

Figure 90: 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

Figure 91: 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

Figure 92: 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

Figure 93: 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

Figure 94: 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

Figure 95: 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 selected half 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 380 shows the bit representation of each word in the palette.

Bit	Name	Description
31	I	Intensity / Not used
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 / Not used
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 380: 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-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)

Table 381: External pad interface signals

Signal name	Type	Description
CLAC	Output	STN AC bias drive or TFT data enable output
CLD[17:0]	Output	LCD panel data
CLLE	Output	Line end signal

Table 381: External pad interface signals

LCD panel signal multiplexing details

The CLLP, CLAC, CLFP, and CLLE signals are common, but the CLD[17:0] bus has seven modes of operation:

- 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 382 shows which CLD[17:0] 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[17]	W18=LCD data bit 17 (02)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[16]	V17=LCD data bit16 (02)	N/A	N/A	N/A	N/A	N/A	N/A
CLD[15]	U16=LCD data bit 15 (02)	N/A	CLSTN[0][1]	N/A	N/A	N/A	MLSTN[0][1]
CLD[14]	Y18=LCD data bit 14 (02)	N/A	CLSTN[1]	N/A	N/A	N/A	MLSTN[1]
CLD[13]	W17=LCD data bit 13 (02)	N/A	CLSTN[2]	N/A	N/A	N/A	MLSTN[2]
CLD[12]	V16=LCD data bit 12 (02)	N/A	CLSTN[3]	N/A	N/A	N/A	MLSTN[3]
CLD[11]	U15=LCD data bit 11 (02) Y16=LCD data bit 11 (02)	N/A	CLSTN[4]	N/A	MLSTN[0][1]	N/A	MLSTN[4]
CLD[10]	Y17=LCD data bit 10 (02) W15=LCD data bit 10 (02)	N/A	CLSTN[5]	N/A	MLSTN[1]	N/A	MLSTN[5]
CLD[9]	W16=LCD data bit 9 (02) V14=LCD data bit 9 (02)	N/A	CLSTN[6]	N/A	MLSTN[2]	N/A	MLSTN[6]
CLD[8]	V15=LCD data bit 8 (02) Y15=LCD data bit 8 (02)	N/A	CLSTN[7]	N/A	MLSTN[3]	N/A	MLSTN[7]
CLD[7]	Y16=LCD data bit 7 (01)	CUSTN[0][1]	CUSTN[0][1]	N/A	N/A	MUSTN[0]	MUSTN[0][1]
CLD[6]	W15=LCD data bit 6 (01)	CUSTN[1]	CUSTN[1]	N/A	N/A	MUSTN[1]	MUSTN[1]
CLD[5]	V14=LCD data bit 5 (01)	CUSTN[2]	CUSTN[2]	N/A	N/A	MUSTN[2]	MUSTN[2]
CLD[4]	Y15=LCD data bit 4 (01)	CUSTN[3]	CUSTN[3]	N/A	N/A	MUSTN[3]	MUSTN[3]
CLD[3]	W14=LCD data bit 3 (01)	CUSTN[4]	CUSTN[4]	MUSTN[0]	MUSTN[0][1]	MUSTN[4]	MUSTN[4]
CLD[2]	Y14=LCD data bit 2 (01)	CUSTN[5]	CUSTN[5]	MUSTN[1]	MUSTN[1]	MUSTN[5]	MUSTN[5]
CLD[1]	V13=LCD data bit 1 (01)	CUSTN[6]	CUSTN[6]	MUSTN[2]	MUSTN[2]	MUSTN[6]	MUSTN[6]
CLD[0]	W13=LCD data bit 0 (01)	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 382: LCD STN panel signal multiplexing

Table 383 shows which CLD[17:0] pins are used to provide the pixel data to the TFT panel for each mode of operation.

External pin	TFT 15 bit
CLD[17]	Blue[4]
CLD[16]	Blue[3]
CLD[15]	Blue[2]
CLD[14]	Blue[1]
CLD[13]	Blue[0]
CLD[12]	Intensity bit
CLD[11]	Green[4]
CLD[10]	Green[3]
CLD[9]	Green[2]
CLD[8]	Green[1]
CLD[7]	Green[0]
CLD[6]	Intensity bit
CLD[5]	Red[4]
CLD[4]	Red[3]
CLD[3]	Red[2]
CLD[2]	Red[1]
CLD[1]	Red[0]
CLD[0]	Intensity bit

Table 383: 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.

	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 384: 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 385 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

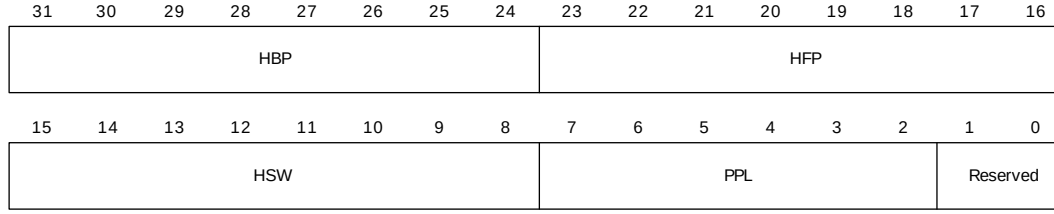
Table 385: LCD registers

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)



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.
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.

Table 386: LCDTiming0 register

Bits	Access	Mnemonic	Reset	Description
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 386: 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. The next table 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 387: Minimum recommended values for STN displays

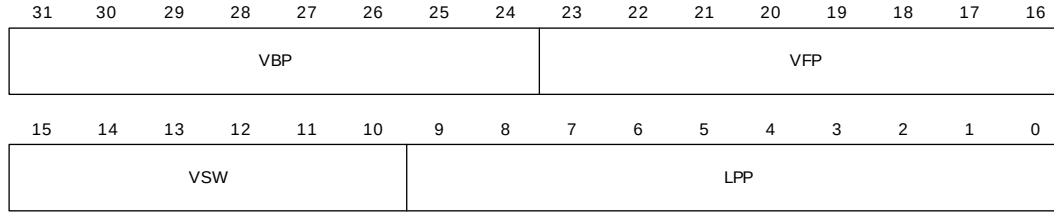
If sufficient time is given at the beginning of the line (for example, HSW is set to 6 and HBP is set to 10), data will not become corrupted for PCD = 4 (minimum value for dual panel mode).

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)



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 388: 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 388: LCDTiming1 register

LCDTiming2

Address: A080 0008

The LCDTiming2 register provides controls for the timing signals.

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 389: 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 NS9360 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 389: 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 550 for more information.

Table 389: LCDTiming2 register

Panel clock divider restrictions

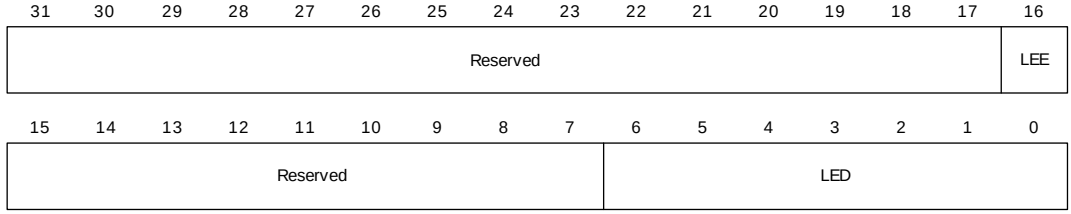
The data path latency forces some restrictions on the usable minimum values for the panel clock divider in STN modes:

- 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.



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 390: 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.

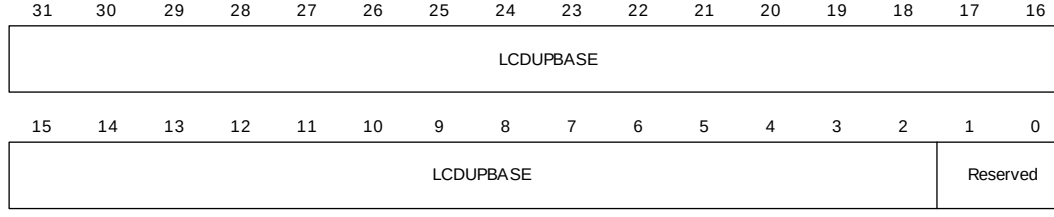
LCDUPBASE is used for these displays:

- 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.

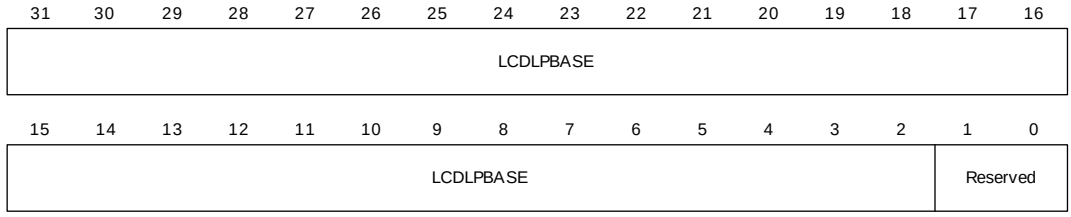
The base address value optionally can be changed mid-frame, to allow double-buffered video displays to be created. These registers are copied to the corresponding current registers at each LCD vertical synchronization, which then cause the LNBU bit (in the LCDStatus register) to be set and an optional interrupt to be generated. The interrupt can be used to reprogram the base address when generating double-buffered video.



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 391: LCDUPBASE register



Register bit assignment

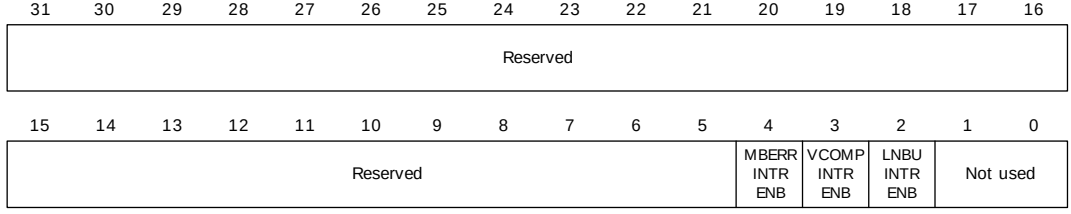
Bits	Access	Mnemonic	Reset	Description
D31:02	R/W	LCDLPBASE	0x00000000	**LCD lower panel base address** Starting address of the lower panel frame data in memory; the address is word-aligned.
D01:00	R/W	Not used	0x0	Read as 0.

Table 392: 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.



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 393: 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
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.)

Table 394: LCDControl register

Bits	Access	Mnemonic	Reset	Description
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 557) 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[17:0] signals disabled (held low). 1 Power gated through to LCD panel and CLD[17:0] signals enabled (active). See "LCD power up and power down sequence support" on page 527 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 533 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
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.

Table 394: LCDControl register

Bits	Access	Mnemonic	Reset	Description
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 Reserved 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 527 for more information about LCD power sequencing.

Table 394: LCDControl register

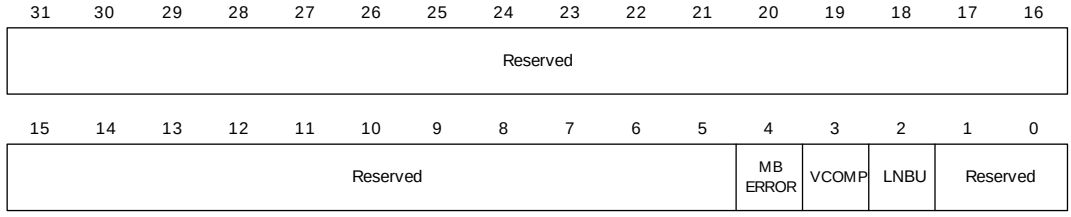
LCDStatus register

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

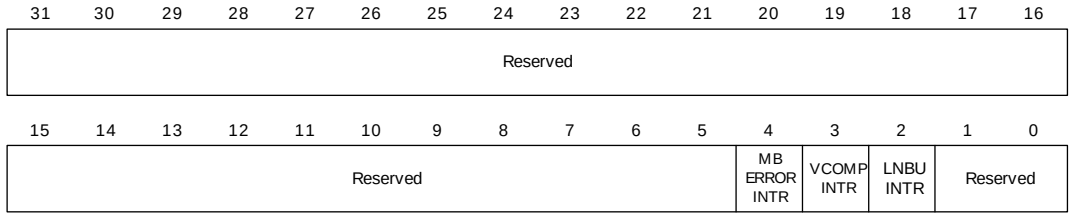
Bits	Access	Mnemonic	Reset	Description
D31:05	N/A	Reserved	N/A	N/A
D04	R/C	MBERROR	0x0	**AHB master bus error status** Set when the AHB master encounters a bus error response from a slave.
D03	R/C	VCOMP	0x0	**Vertical compare** Set when one of the four vertical regions, selected by the LCDControl register, is reached. (See LcdVcomp in "LCDControl register" on page 554).
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 395: 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

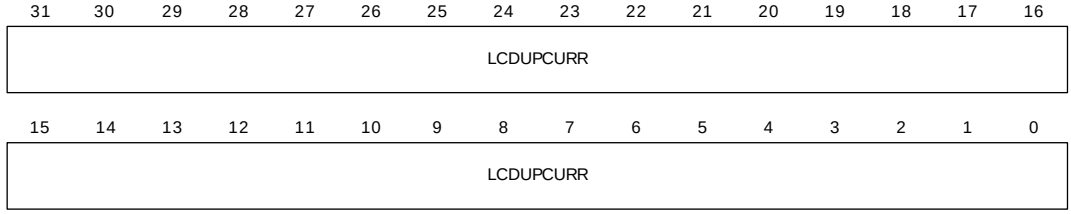
Bits	Access	Mnemonic	Reset	Description
D31:05	N/A	Reserved	N/A	N/A
D04	R	MBERRORINTR	0x0	AHB master bus error interrupt status bit.
D03	R	VCOMPINTR	0x0	Vertical compare interrupt status bit.
D02	R	LNBUINTR	0x0	LCD next base address update interrupt status bit.
D01:00	N/A	Reserved	N/A	N/A

Table 396: LCDInterrupt register

LCDUPCURR and LCDLPCURR

Address: A080 0028 and A080 002C

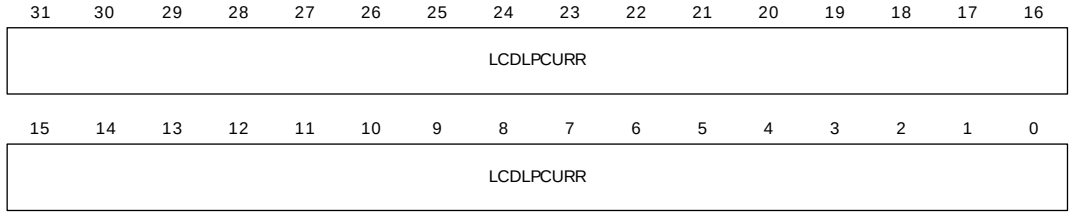
The LCDUPCURR and LCDLPCURR registers contain an approximate value of the upper and lower panel data DMA addresses when read. The registers can change at any time, and therefore can be used only as a mechanism for coarse delay.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	LCDUPCURR	X	LCD upper panel current address value.

Table 397: LCDUPCURR register



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R	LCDLPCURR	X	LCD lower panel current address value.

Table 398: LCDLPCURR register

LCDPalette register

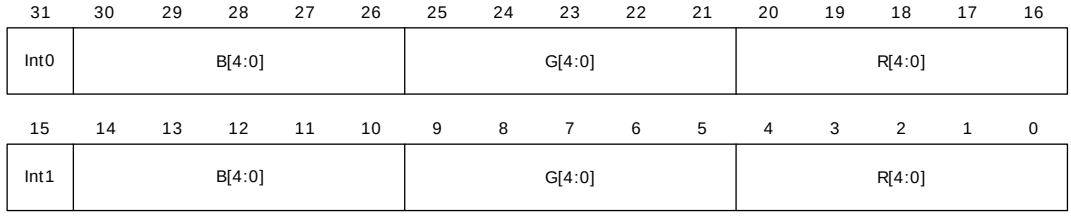
Address A080 0200 – 03FC

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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	Int0	N/A	**Intensity / Not used** 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 399: 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 399: 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 NS9360 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 96 shows the structure of the serial module.

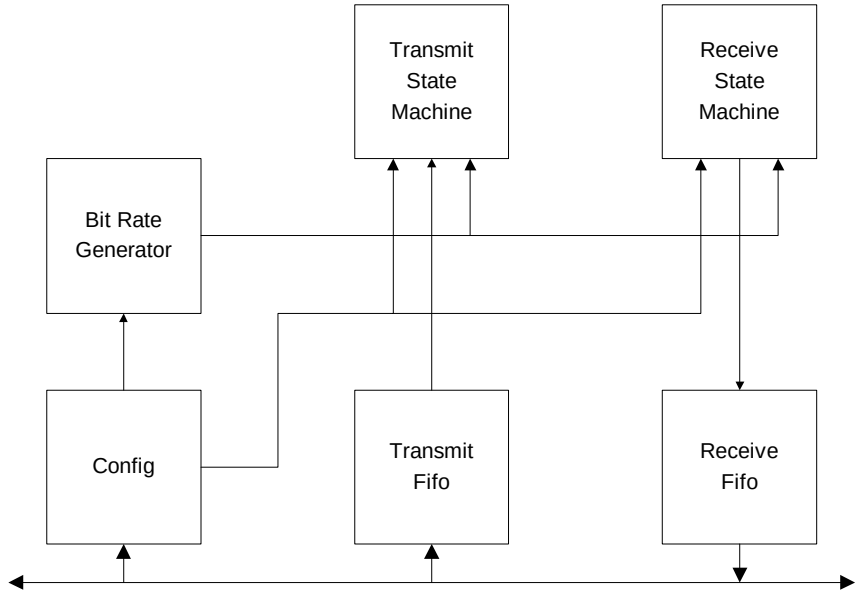


Figure 96: Serial Module structure

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.

The next table describes all possible clock reference sources used by the bit-rate generator.

Name	Description
X1_SYS_OSC/2	The frequency of the external crystal oscillator divided by 2.
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 400: 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.

An inversion function is included in each general purpose I/O pin (see the discussion of GPIO configuration register options) to create the active LOW and inactive HIGH signal levels for the UART control signals (RI, DCD, DSR, DTR, CTS, and RTS). Enable the inversion function for each of the UART signals, as needed.

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 580) 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 574).

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 580).

- 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 580) 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 NS9360 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. The next table 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 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

Table 401: Serial channel B & A configuration registers

Address	Description
9020 0074	Channel A Flow Control register
9020 0078	Channel A Flow Control Force register

Table 401: 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. The next table shows a single, two-channel address map for serial controllers C and D.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

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

Table 402: Serial channel C & D configuration registers

Address	Description
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

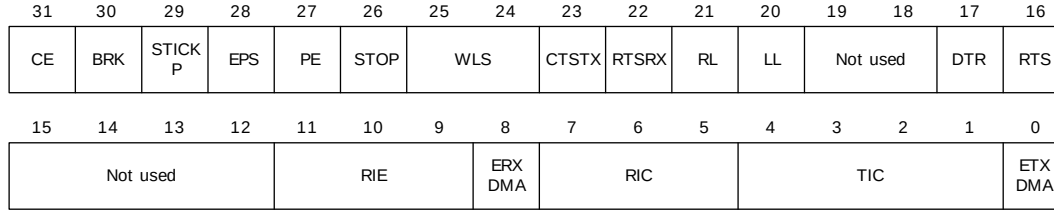
Table 402: Serial channel C & D configuration registers

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.



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, 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 403: 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.
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 403: 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 403: 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] Not used — Set to zero
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.

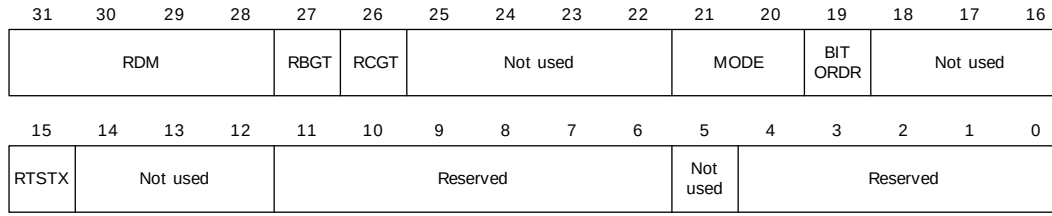
Table 403: Serial Channel B/A/C/D Control Register A

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 404: 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 404: 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			RBC PEND	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. 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 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 405: 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 and the timeout value defined in the Receive Character GAP Timer register 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.
25:23	R	Not used	0x0	This field is always read as 0x0.
D22	R	RBCPEND	0	**Receive buffer closed pending** Set when the receive status FIFO is not empty, indicating that there is at least one receive buffer close event in the status FIFO.
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.

Table 405: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D18	R	RI	0	**Ring indicator** 0 Inactive 1 Active Indicates the current state of the EIA ring indicator signal.
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 405: 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 405: 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	RW1TC	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 two more lines of data.
D07	RW1TC	DCDI	0	**Change in DCD** Indicates a state change in the EIA data carrier detect signal. The DCDI status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. Write a 1 to this bit to clear it.
D06	RW1TC	RII	0	**Change in RI** Indicates a state change in the EIA ring indicator signal. The RII status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Status Register A. Write a 1 to this bit to clear it.

Table 405: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D05	RW1TC	DSRI	0	**Change in DSR** Indicates a state change in the EIA data set ready signal. The DSRI status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. Write a 1 to this bit to clear it.
D04	RW1TC	CTSI	0	**Change in CTS** Indicates a state change in the EIA clear to send signal. The CTSI status condition can be programmed to generate an interrupt by setting the corresponding IE bit in Serial Channel Control Register A. Write a 1 to this bit to clear it.
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.

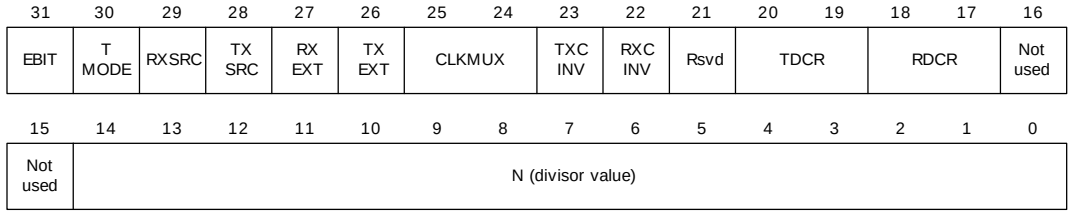
Table 405: Serial Channel B/A/C/D Status Register A

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.



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 406: 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 406: 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/2 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. Note: If option 10 or 11 is selected, the frequency of BCLK must be at least four times greater than the external clock.
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 406: 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 407 for examples.

Table 406: Serial Channel B/A/C/D Bit-rate register

The next table shows 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.

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 407: Bit-rate examples for X1_SYS_OSC/2

Serial Channel B/A/C/D FIFO Data register

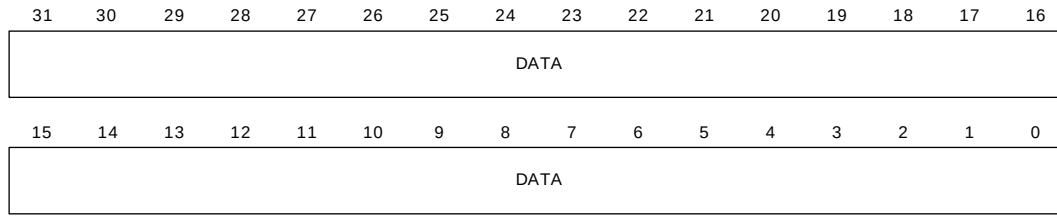
Address: 9020 0010 / 0050

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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	DATA	0x00000000	Serial channel FIFO data field.

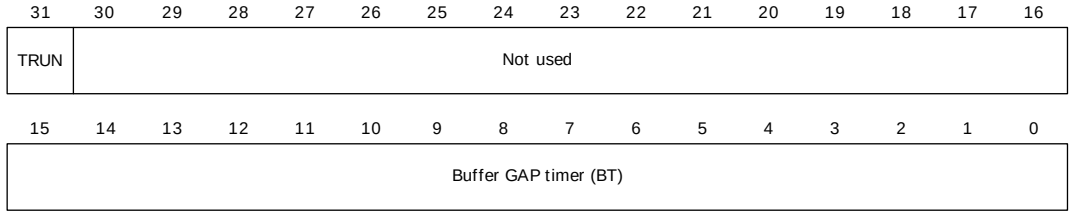
Table 408: Serial Channel B/A/C/D FIFO Data register

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.



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.

Table 409: Serial Channel B/A/C/D Receive Buffer GAP Timer

Bits	Access	Mnemonic	Reset	Description
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 400 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

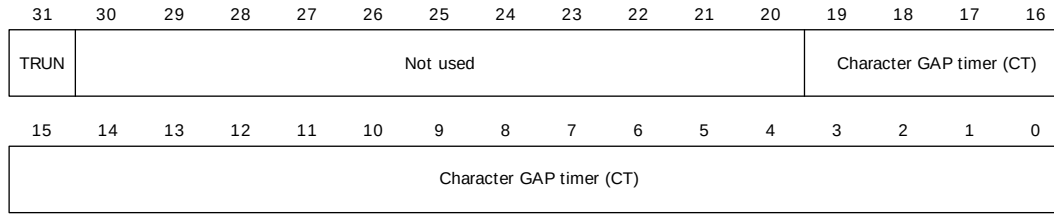
Table 409: Serial Channel B/A/C/D Receive Buffer GAP Timer

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 0.27 us 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.



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.
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 / 2) ■ 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.

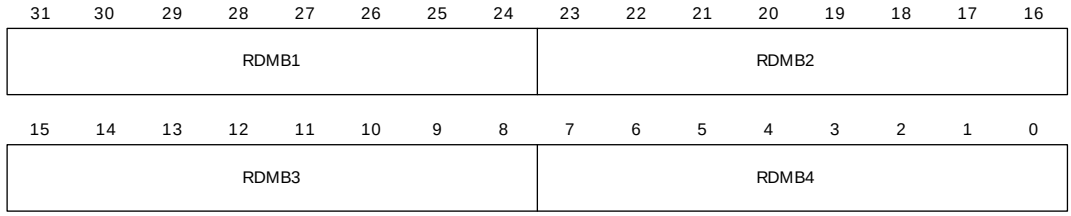
Table 410: Serial Channel B/A/C/D Receive Character GAP Timer

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.



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

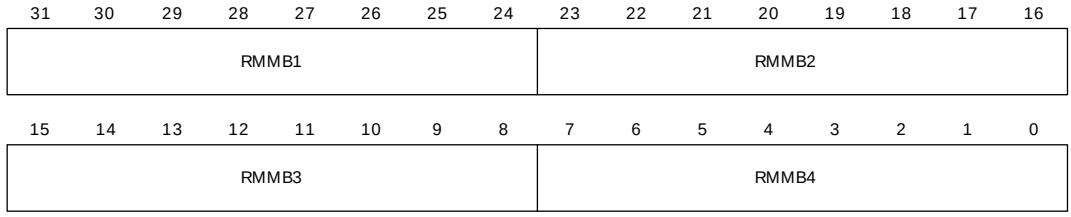
Table 411: Serial Channel B/A/C/D Receive Match register

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

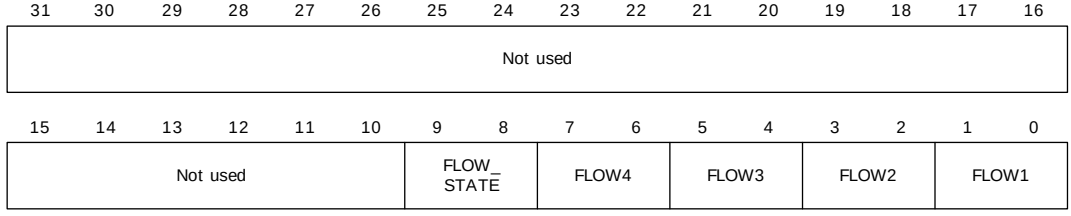
Table 412: Serial Channel B/A/C/D Receive Match MASK register

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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:10	R	Not used	0x000000	Always read as 0x000000.
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 and RMMB4.
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 and RMMB3.
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 and RMMB2.

Table 413: Serial Channel B/A/C/D Flow Control register

Bits	Access	Mnemonic	Reset	Description
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 and RMMB1.

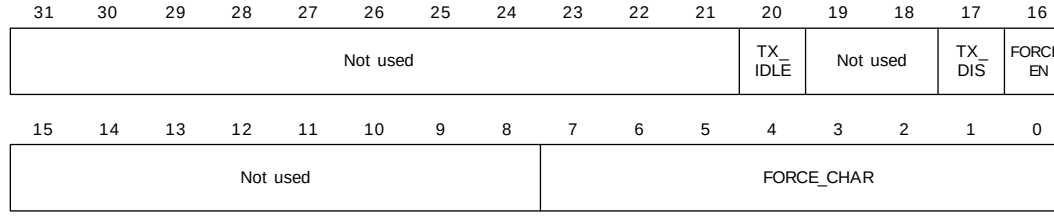
Table 413: Serial Channel B/A/C/D Flow Control register

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).

Table 414: Serial Channel B/A/C/D Flow Control Force register

Bits	Access	Mnemonic	Reset	Description
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.
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 414: Serial Channel B/A/C/D Flow Control Force register

Serial Control Module: SPI

C H A P T E R 1 4

The NS9360 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: 11.07 Mbps
  - SPI slave: 5.5 Mbps
- 32-byte TX FIFO
- 32-Byte RX FIFO

Figure 97 shows the structure of the serial module.

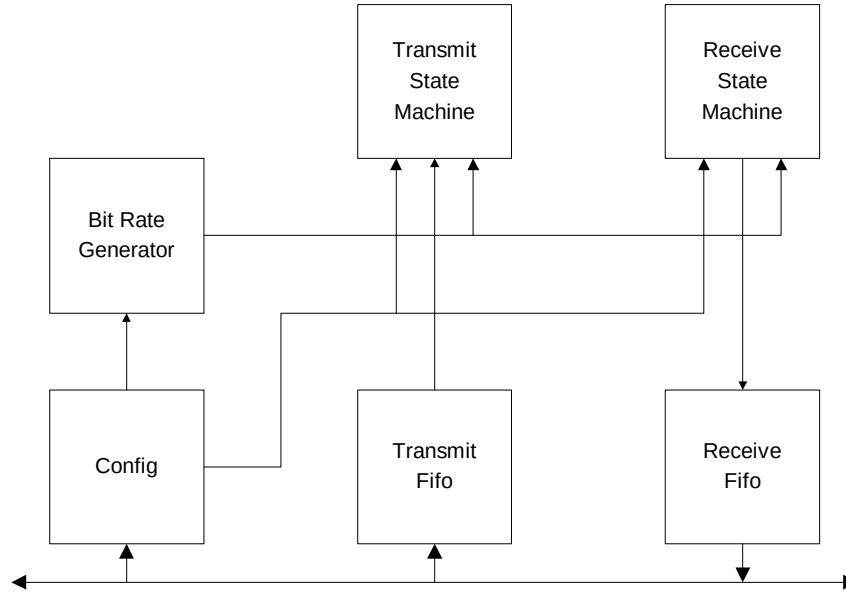


Figure 97: Serial module structure

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.

The next table 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 415: Bit-rate generation clock sources

SPI mode

The NS9360 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 603) 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. The next table 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	1	1	1	Low	Falling	Rising
2	0	1	1	High	Falling	Rising
3	0	0	0	High	Rising	Falling

Table 416: 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 (see "Serial Channel B/A/C/D FIFO Data register," beginning on page 623).

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 615) 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 610).

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 615).

- 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 615) 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 NS9360 ASIC. The configured speed for the internal PLL defines the BCLK rate; for SPI, the serial port maximum rate is BCLK/4.

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. The next table 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 417: 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 417: 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. The next table shows a single, two-channel address map for serial controllers C and D.

All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

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

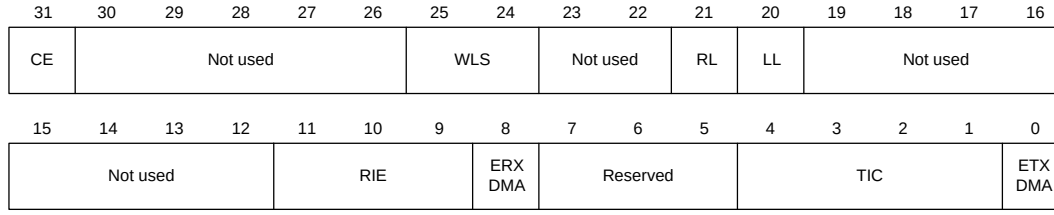
Table 418: Serial channel C & D configuration registers

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.



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, 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** Must be set to 11, 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 419: 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 419: 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 419: Serial Channel B/A/C/D Control Register A

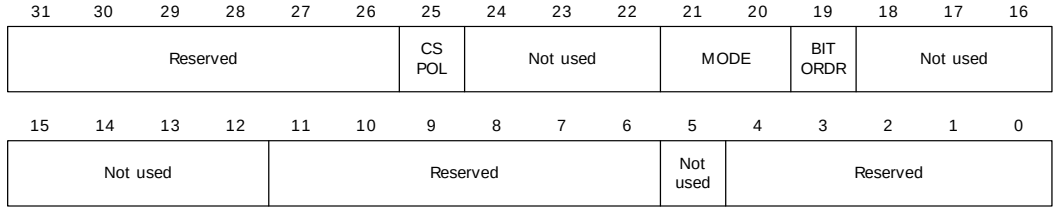
Serial Channel B/A/C/D Control Register B

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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	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 420: 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

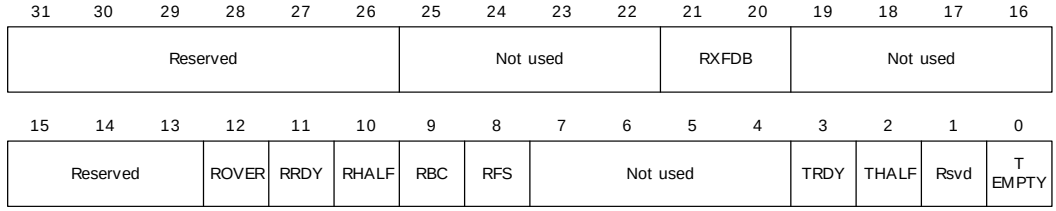
Table 420: Serial Channel B/A/C/D Control Register B

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.



Bits	Access	Mnemonic	Reset	Description
D31:26	N/A	Reserved	N/A	N/A
25:23	R	Not used	0x0	This field is always read as 0x0.
D22	R	RBCPEND	0	**Receive buffer closed pending** Set when the receive status FIFO is not empty, indicating that there is at least one receive buffer close event in the status FIFO.
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.

Table 421: Serial Channel B/A/C/D Status Register A

Bits	Access	Mnemonic	Reset	Description
D19:16	R	Not used	0	Always write as 0.
D15:13	N/A	Reserved	N/A	N/A
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. In 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 421: 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 two more lines 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 421: 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.

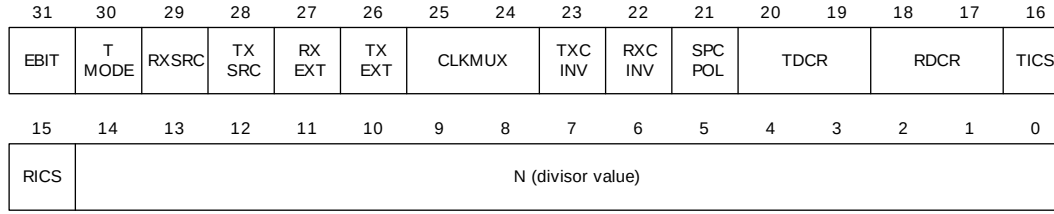
Table 421: Serial Channel B/A/C/D Status Register A

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.



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 modes, 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 modes, 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 modes, set to 0 for disable.

Table 422: 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, set to 1 to enable. For SPI slave, 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 415, "Bit-rate generation clock sources," on page 603 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 422: 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 422: 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 422: 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; the bit-rate is determined by the incoming clock edges.

Table 422: Serial Channel B/A/C/D Bit-rate register

Serial Channel B/A/C/D FIFO Data register

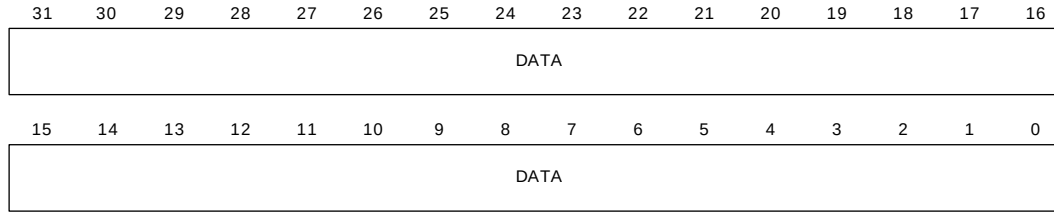
Address: 9020 0010 / 0050
9030 0010 / 0050

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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:00	R/W	DATA	0x00000000	Serial channel FIFO data field.

Table 423: Serial Channel B/A/C/D FIFO Data register

IEEE 1284 Peripheral Controller

CHAPTER 15

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 660).
- **External transceivers.** The data flow direction control is provided using GPIO pin 43 and is under hardware control.

Overview

Figure 98 shows the block diagram of the IEEE 1284 peripheral port control module.

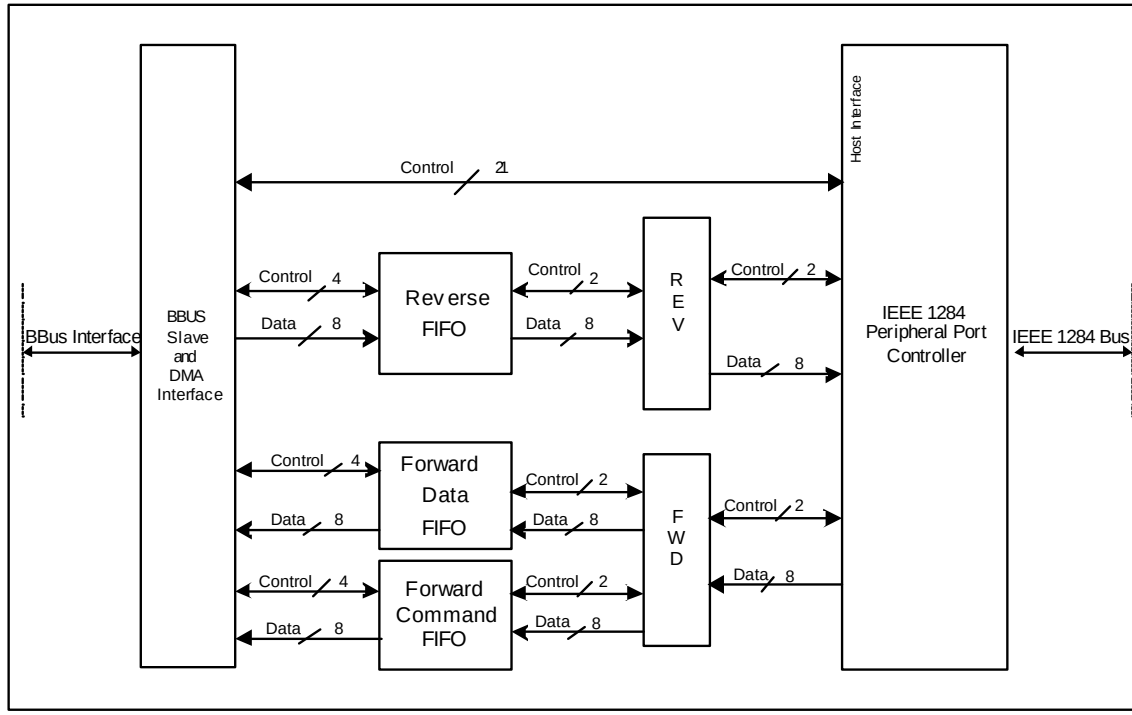


Figure 98: IEEE 1284 peripheral port control module

Note: Traffic direction in the IEEE 1284 is classified as either forward or reverse. The forward direction is equivalent to NS9360 receive. Similarly, the reverse direction is equivalent to NS9360 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 99 shows the timing relationship on the port interface.

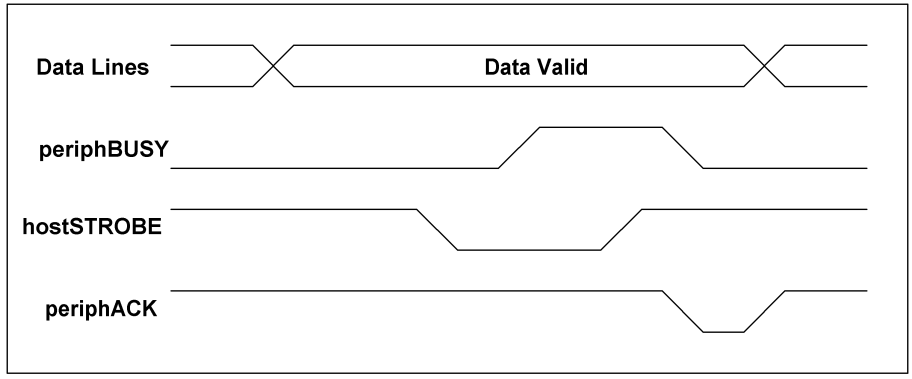


Figure 99: Compatibility mode data transfer cycle

Warning: The nFault and pError signals are supposed to indicate error conditions to the host while the IEEE 1284 interface operates in compatibility mode. These signals cannot be controlled by software, however, while the hardware is configured to automatically negotiate and/or transfer data, which is the normal mode of operation. Therefore, these signals cannot be used to report error conditions in compatibility mode as required by the IEEE 1284 specification.

To avoid problems, do not use these signals in compatibility mode. instead, have the printer firmware and printer driver software on the host use the bidirectional communications channels to report the printer's status.

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 100 shows the timing relationship on the port interface.

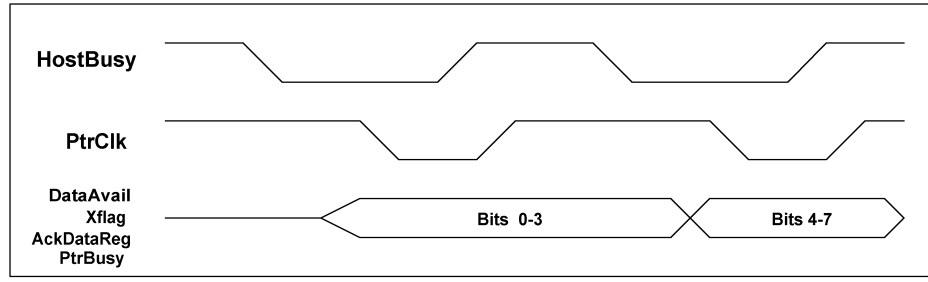


Figure 100: Nibble mode data transfer cycles

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 101 shows the timing relationship on the port interface.

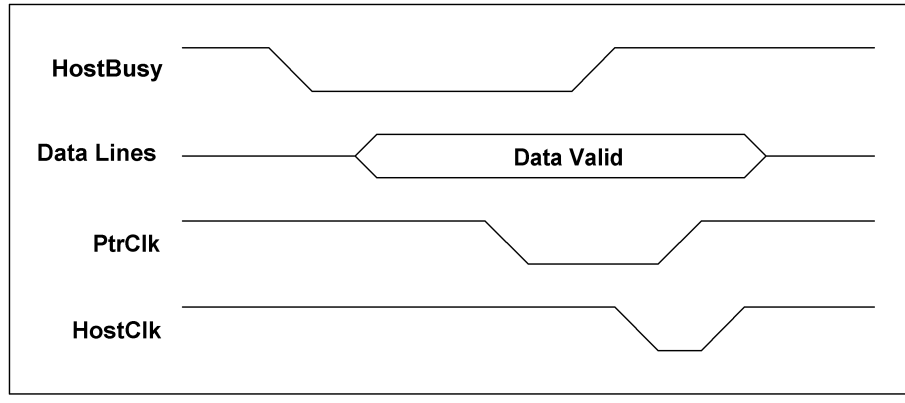


Figure 101: IEEE 1284 byte mode data transfer cycle

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. NS9360 uses RLE decoding to enable large raster images with large strings of identical data to be transferred to system memory.

Forward transfer cycles

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 7 is 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.

Figure 102 shows a data cycle followed by a command cycle.

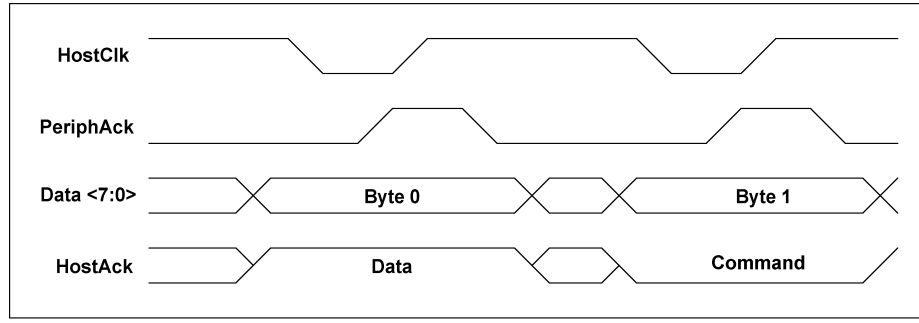


Figure 102: ECP mode forward transfer cycles

▶ **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 103 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.

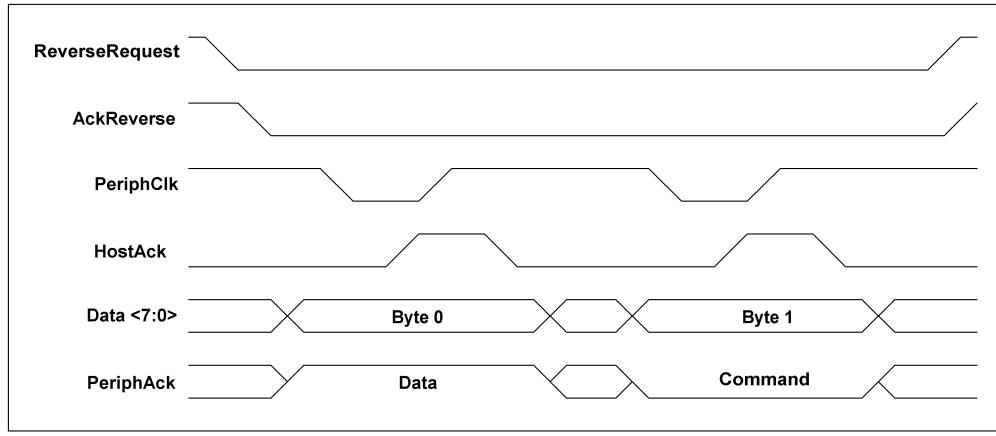


Figure 103: ECP reverse channel transfer cycles

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 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 NS9360. 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 NS9360.

Because the NS9360 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 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.

The next table 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.
0011 0100	Request device ID using ECP mode with RLE	Receive device ID with ECP data compression.
0000 0010	Reserved	Reserved

Table 424: Extensibility byte values

Extensibility byte	Definition	Description
0000 0001	Byte mode reverse channel transfer	
0000 0000	Nibble mode reverse channel transfer	

Table 424: Extensibility byte values

The NS9360 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.

Supporting Windows plug-n-play through the NS9360 IEEE 1284 port

The WIndows plug-n-play driver uses nibble ID mode to identify the type of peripheral attached to a parallel port. The Windows host driver negotiates into nibble ID mode and then attempts to read the device ID from the peripheral. When a host negotiates into nibble ID mode with the NS9360, the NS9360 immediately reports whether it has data ready to send in the transmit FIFO:

- If the TX FIFO contains data, nDataAvail(nFault) is set low.
- If the TX FIFO is empty, nDataAvail(nFault) is set high.

The same FIFO is used for transmitting the device ID information as well as other reverse data. The software on the NS9360 does not normally preload data into the TX FIFO, as the software does not know into which mode the host will negotiate and therefore does not know which data to put into the FIFO. The NS9360 allows the software to wait until the host negotiates into a reverse data mode, at which time it knows which data to send to the host. The correct data is then loaded into the TX FIFO. When the Windows plug-n-play driver negotiates into nibble ID mode, the driver checks the state of the nDataAvail(nFault) line to determine whether the peripheral has a device ID string.

This situation creates a "race" condition between the software and the Windows plug-n-play driver:

- The TX FIFO must be loaded with data before the Windows driver checks the state of nDataAvail(nFault).
- The Windows driver checks nDataAvail(nFault) immediately upon negotiating into nibble ID mode.

A delay occurs between the time the host negotiates into nibble ID mode and when software on the NS9360 responds to the interrupt and loads the transmit FIFO with the ID string. The interface signals that no data is available until after software loads the ID string. When the Windows plug-n-play driver polls the NS9360 peripheral device to check for a device ID string and finds that there is no data available, the driver terminates nibble ID mode before the software can load the TX FIFO with the correct data. This prevents Windows plug-n-play from running.

To avoid this problem, preload the device ID data into the transmit FIFO. When the Windows plug-n-play driver negotiates into nibble ID mode, the NS9360 reports that device ID data is available by setting nDataAvail(nFault) low. The Windows driver then reads the ID string and enables Windows plug-and-play to automatically identify and install the peripheral.

Be aware that the device ID string will be sent to the host whenever the host negotiates from compatibility mode into any reverse data mode. To avoid problems, you can do any of the following:

1. If the peripheral sends the host only the device ID as reverse data, there is no issue.
2. If the peripheral uses reverse data mode only occasionally, the device driver on the host machine should be written to ignore the device ID string sent when a reverse data mode is negotiated.
3. If the peripheral transfers a lot of reverse data, use ECP mode for both forward and reverse data transfers. The device ID string can be loaded into the TX FIFO when the host negotiates to compatibility mode, so the device ID will always be sent when Windows plug-n-play negotiates from compatibility mode into nibble ID mode. When a job starts, the host driver should negotiate into ECP mode and use ECP forward mode to transfer data from the host to the peripheral and ECP reverse mode to transfer data from the peripheral to the host.

   Only a single device ID string (loaded into the TX FIFO when the port was in compatibility mode) is sent the first time ECP reverse mode is used. Switching between ECO forward mode and ECP reverse mode does not cause additional device ID strings to be sent.

Note: The NET+OS parallel driver determines, using the PCM_PRELOAD_DEVICE_ID ioctl, whether the device ID will be preloaded. If you are a NET+OS user, see your NET+OS online documentation.

Abnormal error handling

An abnormal error occurs when the host (or peripheral) is out of compliance with the IEEE 1284 standards specification. If the host driver experiences a catastrophic event (for example, a power loss or software application crash), the 1284 interface cannot respond properly to communications from the host.

If the abnormal error occurs within compatibility mode, the interface does respond to new negotiation; the nACK signal toggles just after event #6 although the pError and nACK signals are not in tandem at event #5.

If the abnormal error occurs outside of compatibility mode, you need to reset the 1284 interface.

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 the next table.

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
9040 0008	FIFO Status	FIFO Status register

Table 425: 1284 Control and Status registers

Address	Register	Description
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	Reserved	
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
9040 016C – 9040 0170		Reserved

Table 425: 1284 Control and Status registers

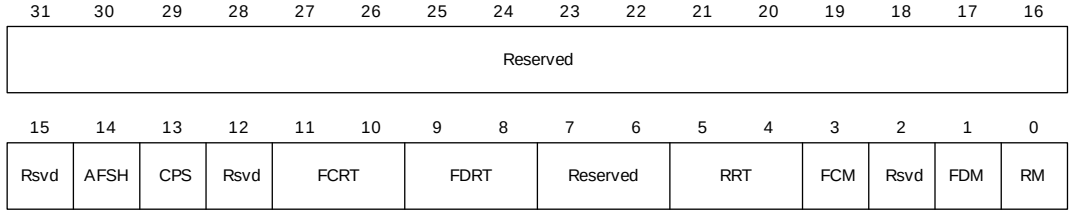
Address	Register	Description
9040 0174	eca	Forward Address register
9040 0178	pha	Core Phase register

Table 425: 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.



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 426: 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 13–16 bytes 10 21–24 bytes 11 25–28 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 426: IEEE 1284 General Configuration register

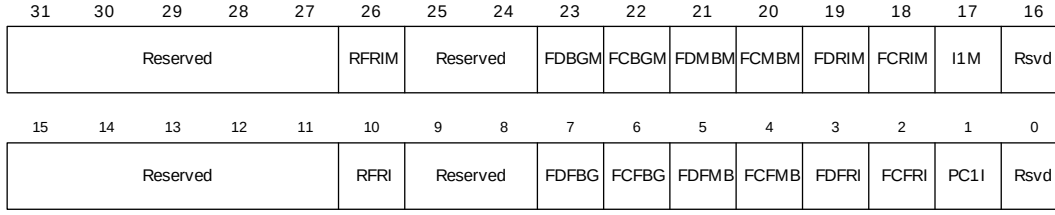
Bits	Access	Mnemonic	Reset	Description
D00	R/W	RM	0x0	**Reverse mode (RevMode)** 0 Direct CPU access 1 DMA control

Table 426: 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.



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 427: 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 427: 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 427: Interrupt Status and Control register

FIFO Status register

Address: 9040 0008

The FIFO Status register allows the CPU to determine the status of all FIFOs in the 1284 module. You can ignore this register when running the 1284 interface in DMA mode.

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 428: 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

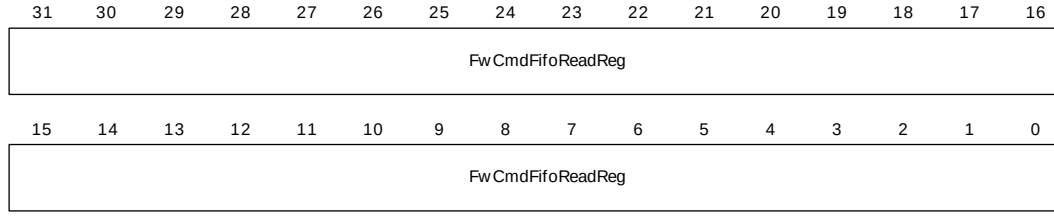
Table 428: FIFO Status register

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 if 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 428: FIFO Status register

Forward Command FIFO Read register

Address: 9040 000C



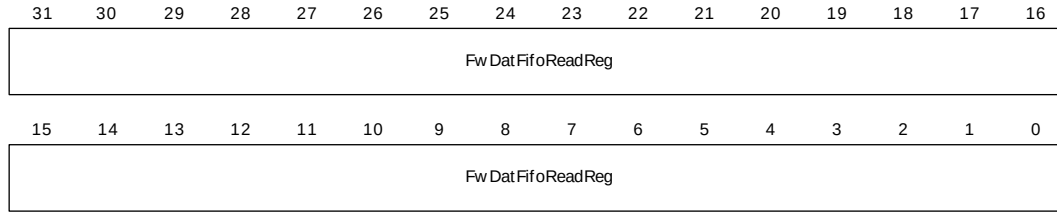
Register bit assignment

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 to determine how many bytes are remaining before issuing the read.

Table 429: Forward Command FIFO Read register

Forward Data FIFO Read register

Address: 9040 0010



Register bit assignment

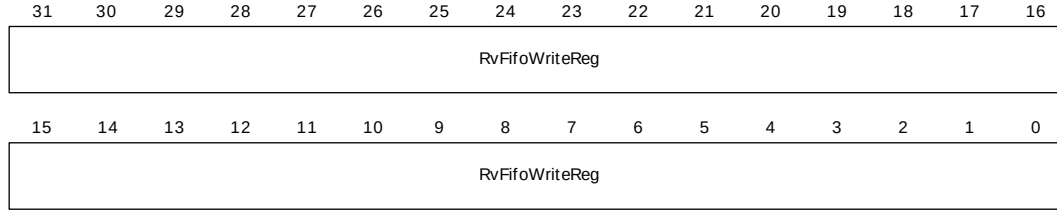
Bits	Access	Mnemonic	Reset	Description
D31:00	R	FwDatFifoReadReg	N/A	Reads up to four bytes from the Forward Data FIFO when in CPU mode. The CPU must read the FIFO Status register to determine how many bytes are remaining before issuing the read.

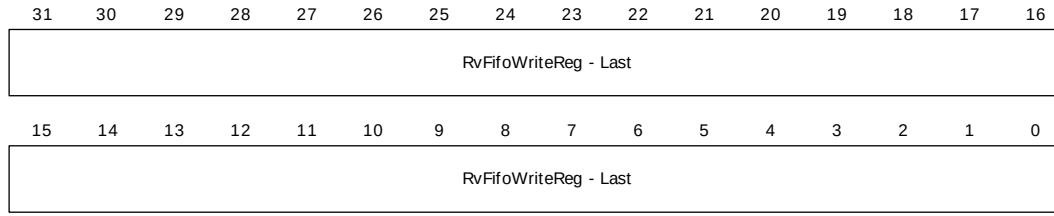
Table 430: Forward Data FIFO Read register

Reverse FIFO Write register/Reverse FIFO Write Register — Last

Address: 9040 001C / 9040 0020

Both registers are 32 bits.





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 431: Reverse Data FIFO Write register/Reverse Data FIFO Write Register — Last

Forward Command DMA Control register

Address: 9040 0024

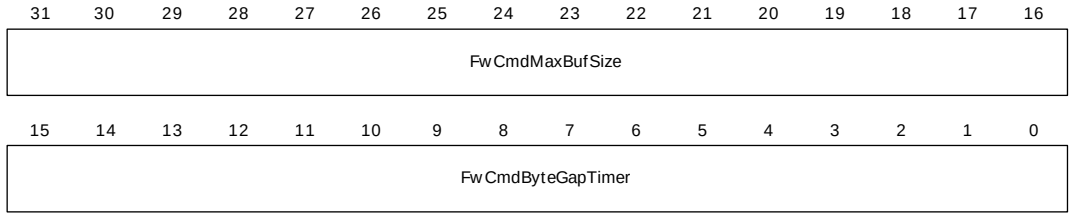
The Forward Command DMA Control register controls when the Forward command DMA buffer is closed, using two components:

- **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:

  1. 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.
  2. 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.
  3. 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	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 432: 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.

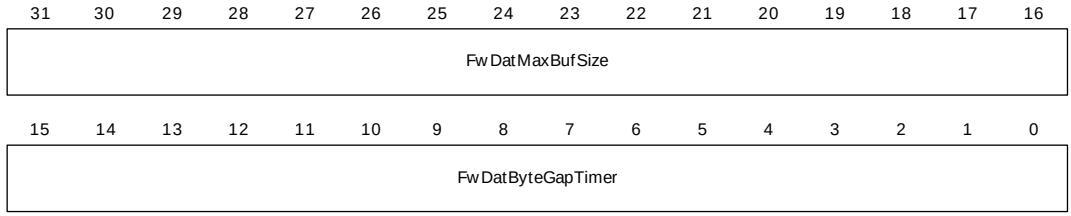
  Note: This counter should not be set to a value greater than the buffer length field value set in the 1284 forward data DMA channel descriptor.

- **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:

  1 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.

2 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.

3 When the data in the FIFO, including the incomplete dwords in Step 1, is output through DMA, the DMA is terminated.



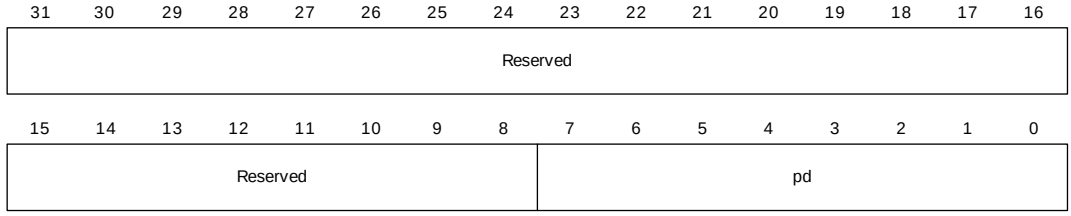
Register bit assignment

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 433: Forward Data DMA Control register

Printer Data Pins register

Address: 9040 0100



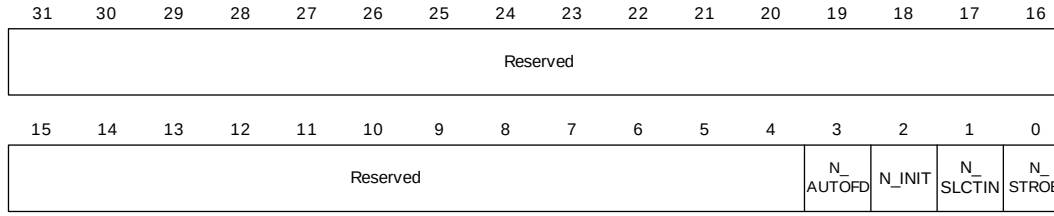
Register bit assignment

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 434: pd — Printer Data Pins register

Port Status register, host

Address: 9040 0104



Register bit assignment

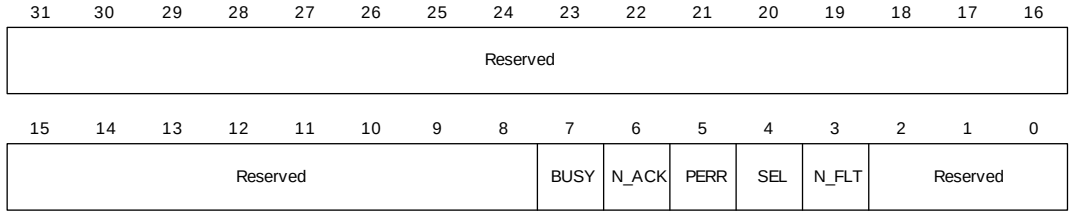
Bits	Access	Mnemonic	Reset	Description
D31:04	N/A	Reserved	N/A	N/A
D03	R	N_AUTOFD	N/A	Allows the CPU to read the status of the host control pins directly. The meaning of each bit varies, depending on whether the mode is compatibility, nibble, byte, or ECP.
D02	R	N_INIT	N/A	
D01	R	N_SLCTIN	N/A	
D00	R	N_STROBE	N/A	

Table 435: 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.



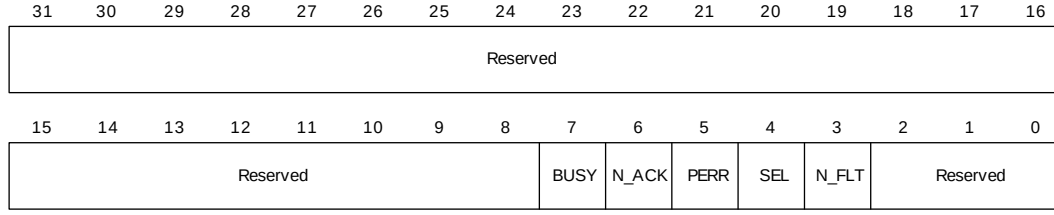
Register bit assignment

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.
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 436: pcr — Port Control register

Port Status register, peripheral

Address: 9040 010C



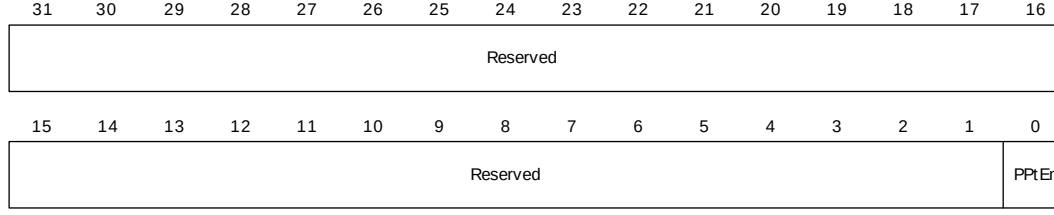
Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07	R	BUSY	0x0	Allows the CPU to read the status of the peripheral control pins directly. The meaning of each bit varies, depending on whether the mode is compatibility, nibble, byte, or ECP.
D06	R	N_ACK	0x0	
D05	R	PERR	0x0	
D04	R	SEL	0x0	
D03	R	N_FLT	0x0	
D02:00	R	Reserved	0x0	N/A

Table 437: 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

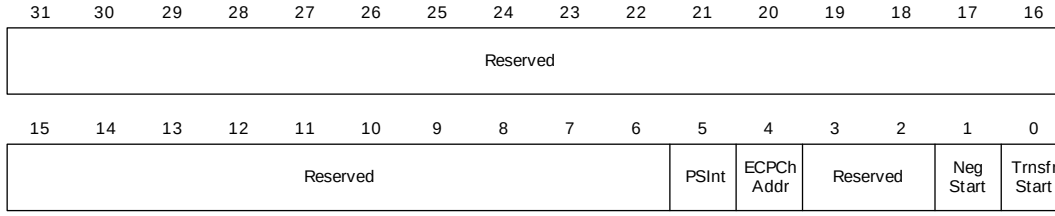
Bits	Access	Mnemonic	Reset	Description
D31:01	N/A	Reserved	N/A	N/A
D00	R/W	PPtEn	0x0	**Printer port enable** 0 Force IEEE 1284 outputs to high impedance 1 Enable normal operation, depending on mode

Table 438: fea — Feature Control Register A

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
D05	R/W	PinSelectInterrupt	0x0	**Pin select interrupt enable** 0 Disable 1 Enable

Table 439: fei — Interrupt Enable register

Bits	Access	Mnemonic	Reset	Description
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 nSTROBE is found during a negotiation (event #4).
D00	R/W	TransferStart	0x0	**Transfer start interrupt enable** 0 Disable 1 Enable This interrupt is triggered when the falling edge of nSTROBE is found while in compatibility mode.

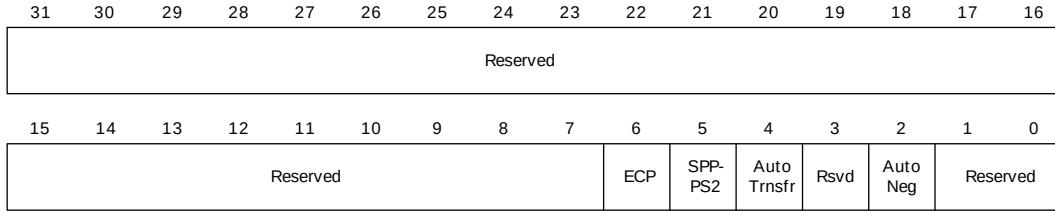
Table 439: fei — Interrupt Enable register

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

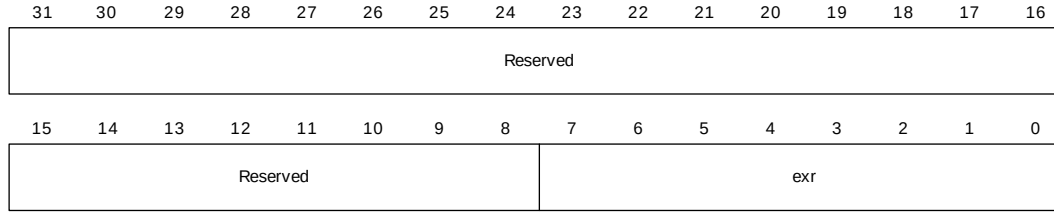
Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	R/W	ECP	0x0	**ECP mode** 0 Disable 1 Enable
D05	R/W	SPP–PS2	0x0	**SPP–PS2 mode** 0 Disable 1 Enable
D04	R/W	AutoTransfer	0x0	**Auto transfer mode** 0 Disable 1 Enable
D03	N/A	Reserved	N/A	N/A
D02	R/W	AutoNegotiate	0x0	**Auto negotiate mode** 0 Disable 1 Enable
D01:00	N/A	Reserved	N/A	N/A

Table 440: fem — Master enable register

Extensibility Byte Requested by Host

Address: 9040 0124

This register is updated when a new negotiation occurs (event #4 of the negotiation process; see the IEEE 1284 standard for more information). The register is also updated with a value of 0x08 when the host terminates the current mode, which places the NS9360 into compatibility mode.



Register bit assignment

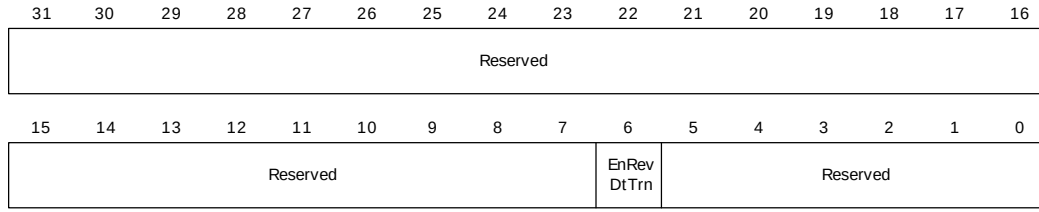
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R/W	exr	0x08	**Extensibility byte** Stores the extensibility byte received from the host.

Table 441: exr — Extensibility Byte Requested by Host register

Extended Control register

Address: 9040 0128

The Extended Control register enables additional core features.



Register bit assignment

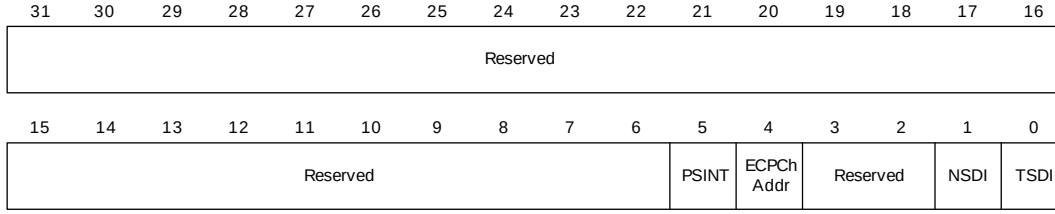
Bits	Access	Mnemonic	Reset	Description
D31:07	N/A	Reserved	N/A	N/A
D06	R/W	Enable reverse data transfers	0x0	0 Disable 1 Enable
D05:00	N/A	Reserved	N/A	N/A

Table 442: 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

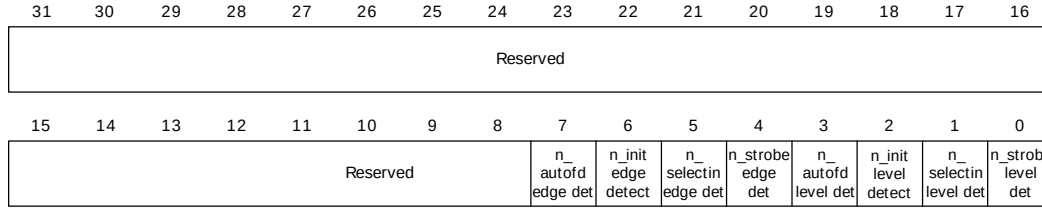
Bits	Access	Mnemonic	Reset	Description
D31:06	N/A	Reserved	N/A	N/A
D05	R	PSINT	0x0	Pin select interrupt
D04	R	ECP channel address	0x0	Channel address update detect interrupt
D03:02	N/A	Reserved	N/A	N/A
D01	R	NSDI	0x0	Negotiation start detect interrupt
D00	R	TSDI	0x0	Transfer start detect interrupt

Table 443: sti — Interrupt Status register

Pin Interrupt Mask register

Address: 9040 0134

The Pin Interrupt Mask register enables IEEE 1284 pin interrupts.



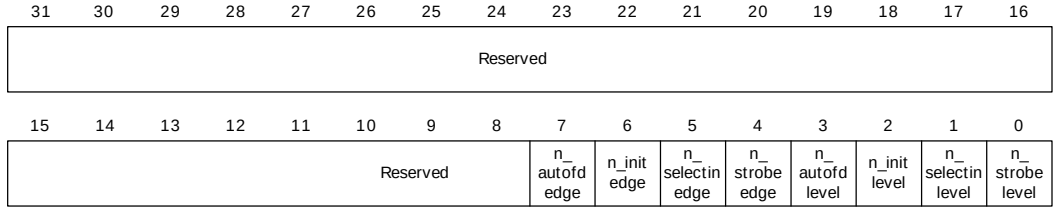
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	0x0	N/A
D07	R/W	n_autofd edge detect	0x0	0 Disable 1 Enable
D06	R/W	n_init edge detect	0x0	
D05	R/W	n_selectin edge detect	0x0	
D04	R/W	n_strobe edge detect	0x0	
D03	R/W	n_autofd level detect	0x0	
D02	R/W	n_init level detect	0x0	
D01	R/W	n_selectin level detect	0x0	
D00	R/W	n_strobe level detect	0x0	

Table 444: msk — Pin Interrupt Mask register

Pin Interrupt Control register

Address: 9040 0138

The Pin Interrupt Control register configures IEEE 1284 pin interrupt edge levels.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07	R/W	n_autofd edge detect	0x0	Must be set to 1 (rising edge)
D06	R/W	n_init edge detect	0x0	
D05	R/W	n_selectin edge detect	0x0	
D04	R/W	n_strobe edge detect	0x0	
D03	R/W	n_autofd level	0x0	0 Low level 1 High level
D02	R/W	n_init level	0x0	
D01	R/W	n_selectin level	0x0	
D00	R/W	n_strobe level	0x0	

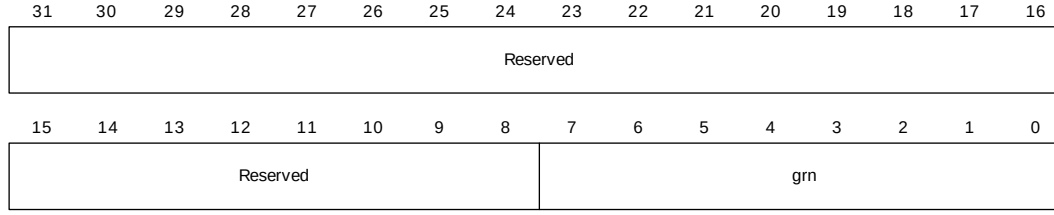
Table 445: pit — Pin Interrupt Control register

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 specification, the peripheral has a T_p (500ns) minimum setup pulse width for some signals. If, for example, the BBus is set to run at 45 MHz (22ns clock period), the Granularity Count register should be set to 23 (0x17) [500ns / 22ns].



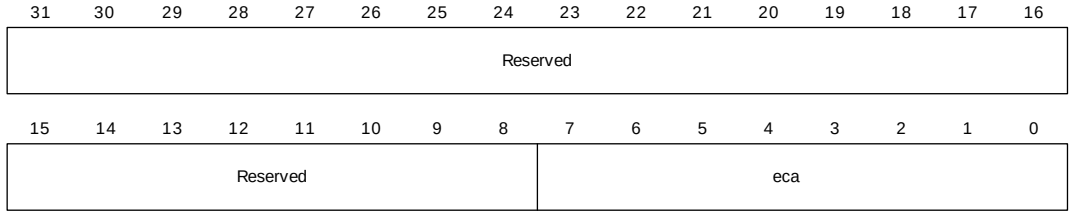
Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0x0	N/A
D07:00	R/W	grn	0x0	**Granularity counter** Determines the number of BBus clock periods between peripheral signal changes on the IEEE 1284 bus.

Table 446: 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.



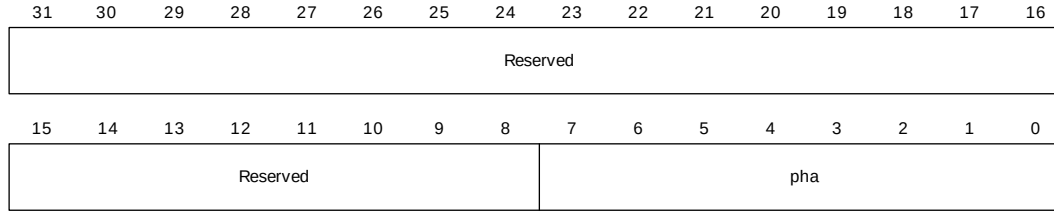
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:08	R	Reserved	0	N/A
D07:00	R	eca	0	Forward address.

Table 447: eca — Forward Address register

Core Phase (IEEE1284) register

Address: 9040 0178



Register bit assignment

Bits	Access	Mnemonic	Reset	Description	
D31:08	R	Reserved	0x0	N/A	
D07:00	R	pha	0x0	0x00	spp 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
				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 448: Core Phase register

USB Host Module

CHAPTER 16

The USB Host module provides a USB 2.0 compliant host interface. The USB Host module in NS9360 supports both full-speed (12Mbps) and low-speed (1.5 Mbps) operation.

Note: You should be familiar with USB 2.0 specifications for both full-speed and low-speed operations before using this module with the NS9360.

Overview

The USB Host module consists of a USB Host controller that conforms to the Open Host Controller Interface (OHCI) specification and a wrapper to interface the module to the rest of the system. The USB Host module interfaces to the internal USB PHY provided by the NS9360.

USB Host module architecture

The USB Host module is comprised of a USB OHCI Host controller and a USB Host Front End (UHFE) block that interfaces the USB OHCI Host controller to the BBus as both a slave and a master. The UHFE also provides control and status registers for managing the USB Host block.

Figure 104 shows the USB Host module architecture.

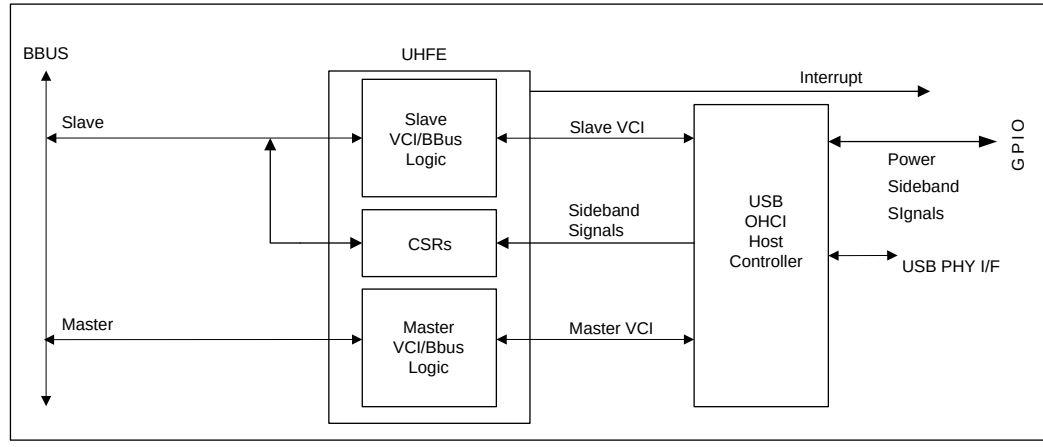


Figure 104: USB Host module architecture

USB OHCI Host controller

The USB OHCI Host controller has these features:

- Support for USB 2.0 full-speed (12 Mbps) operation
- Support for USB 2.0 low-speed (1.5 Mbps) operation
- OHCI Host support
- Slave interface for configuration and master interface for command/data transfer
- Error reporting

Slave and master VCI buses

The USB OHCI Host controller interfaces to the BBus through separate slave and master VCI (virtual component interface) buses.

The slave VCI is used only for accessing the OHCI registers within the USB Host controller.

The master VCI is used to:

- Fetch data transfer descriptors from system memory
- Execute data transfers to and from system memory
- Update transfer status to the Host Controller Communications Area (HCCA) in system memory

The master VCI bus accesses system memory in 4-word bursts for all data reads, data writes, and descriptor reads, and provides a 4-deep FIFO in each direction to accommodate these bursts.

USB PHY interface

The USB OHCI Host controller has an interface to the NS9360's internal USB PHY, and also provides several sideband signals that provide status back to the UHFE control and status registers as well as the GPIO.

USB Host Front End block

The USB Host Front End (UHFE) block interfaces the USB OHCI Host controller to the BBus as both a slave and a master, providing the control logic that bridges both VCI interfaces to the BBus. The UHFE also contains control and status registers that manage the USB Host module.

For big endian applications, the UHFE byte swaps all transfer descriptors read by the USB OHCI Host controller block to compensate for the byte swapping done by the BBus bridge.

The endian configuration for the USB Host module defaults to the strapping value on gpio[44] after reset and powerup. After powerup and reset, the endian configuration is controlled using the USBHST bit in the Endian Configuration register (in the BBus Utility chapter).

Control and status registers

The control and status registers in the USB OHCI Host controller and UHFE blocks are accessed through the BBus slave interface. Only single, 32-bit transfers are supported. No burst accesses to these registers are allowed.

Interrupt

The USB Host module interrupt is connected directly to the NS9360 interrupt controller. The interrupt is also connected to the BBus bridge, where it sets the USBHST bit in the BBus Bridge Interrupt Status register.

GPIO signals

Two sideband signals from the USB OHCI block are connected to the NS9360 GPIO:

- USB_PWR is connected to gpio[17]. This signal is an output from the USB Host module and controls a USB power switch in the system.
- USB_OVR is connected to gpio[16]. This signal is an input to the USB Host module and indicates that an overcurrent condition exists on the USB bus.

USB Host module registers

The USB Host module registers are located at base address 0x9080_0000. Table 449 describes the USB Host address map.

Address range	Register space
0x90800000–0x90800FFF	UHFE Control and Status
0x90801000–0x90801FFF	USB OHCI Host controller

Table 449: USB Host address map

UHFE control and status registers

Table 450 provides the addresses for the UHFE control and status registers. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

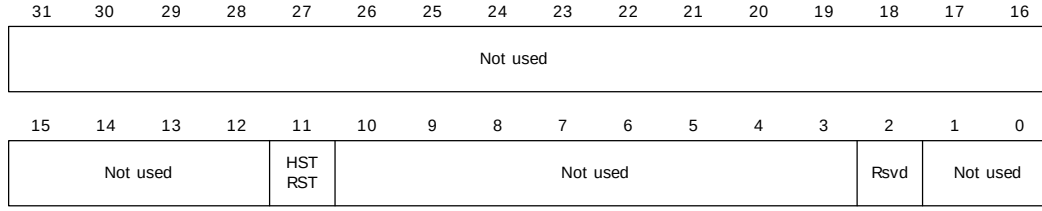
Address	Register
9080 0000	UHFE Control and Status register
9080 000C	UHFE Interrupt Enable register
9080 0010	UHFE Interrupt Status register

Table 450: UHFE control and status register address map

UHFE Control and Status register

Address: 9080 0000

The UHFE Control and Status register contains USB Host front-end control and status information.



Register bit assignment

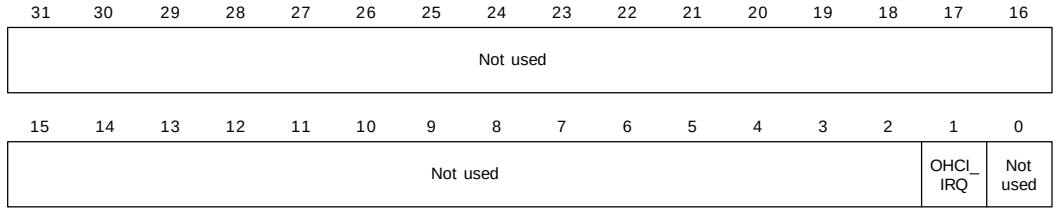
Bits	Access	Mnemonic	Reset	Description
D31:12	R	Not used	0x00000	Always read as 0.
D11	R	HSTRST	1	**Host reset** Read-only field that provides the Host block reset status. 0 The Host block is operational. 1 The Host block is in reset.
D10:03	R	Not used	0x00	Always read as 0.
D02	N/A	Reserved	N/A	N/A
D01:00	R	Not used	00	Always read as 0.

Table 451: UHFE Control and Status register

UHFE Interrupt Enable register

Address: 9080 000C

The UHFE Interrupt Enable register contains UHFE interrupt enable information. *This interrupt is enabled by writing a 1 and disabled by writing a 0.*



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	R	Not used	0x00000000	Always read as 0.
D01	RW	OHCI_IRQ	0	**OHCI_IRQ** Generates an interrupt when the OHCI_IRQ bit in the Host Interrupt Status register is asserted.
D00	R	Not used	0	Always read as 0.

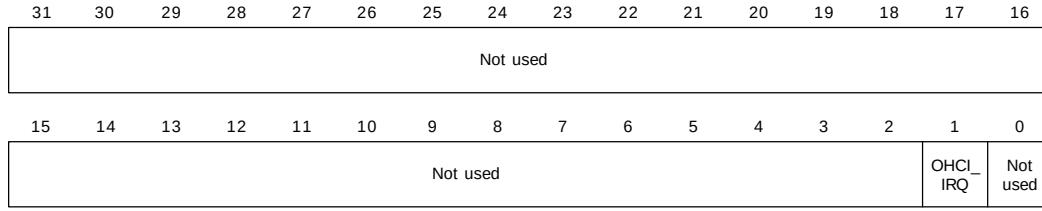
Table 452: UHFE Interrupt Enable register

UHFE Interrupt Status register

Address: 9080 0010

The UHFE Interrupt Status register contains UHFE interrupt status information. *This status bit is active high (1) and is cleared by writing a 1 (RW1TC) to the appropriate field.*

For diagnostics, the status bit can be set to 1 by writing a 1 when the bit is set to 0.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:02	R	Not used	0x00000000	Always read as 0.
D01	RW1TC	OHCI_IRQ	0	**OHCI_IRQ** Asserted when the USB OHCI Host controller generates an interrupt.
D00	R	Not used	0	Always read as 0.

Table 453: UHFE Interrupt Status register

USB OHCI Host controller registers

The USB OHCI Host controller registers are defined in the Open HCI specification for USB. All references to *HC* refer to the USB OHCI Host controller in the NS9360.

Reserved bits

The Host Controller Driver (HCD) should always preserve the value(s) of the reserved field. When a R/W (read/write) 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 OHCI Host controller register address map

Table 454 provides the addresses of the USB OHCI Host controller registers. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register
9080 1000	HcRevision register
9080 1004	HcControl register
9080 1008	HcCommandStatus register
9080 100C	HcInterruptStatus register
9080 1010	HcInterruptEnable register
9080 1014	HcInterruptDisable register
9080 1018	HcHCCA (Host Controller Communications Area) register
9080 101C	HcPeriodCurrentED (Endpoint Descriptor) register
9080 1020	HcControlHeadED register
9080 1024	HcControlCurrentED register

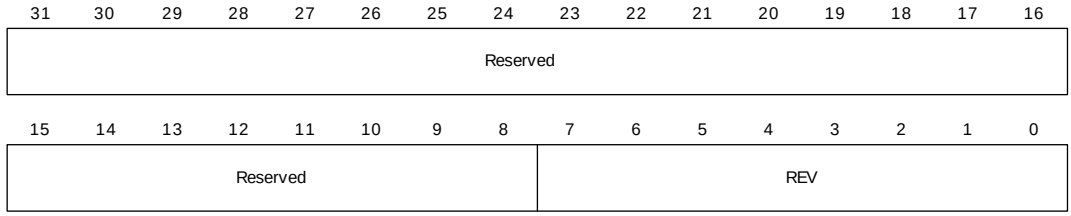
Table 454: USB OHCI Host controller register address map

Address	Register
9080 1028	HcBulkHeadED register
9080 102C	HcBulkCurrentED register
9080 1030	HcDoneHead register
9080 1034	HcFmInterval (Fm=Frame)
9080 1038	HcFmRemaining register
9080 103C	HcFmNumber register
9080 1040	HcPeriodicStart register
9080 1044	HcLSThreshold register
9080 1048	HcRhDescriptorA register (Rh=Root hub)
9080 104C	HcRhDescriptorB register
9080 1050	HcRhStatus register
9080 1054	HcRhPortStatus[1] register

Table 454: USB OHCI Host controller register address map

HCRevision register

Address: 9080 1000



Register bit assignment

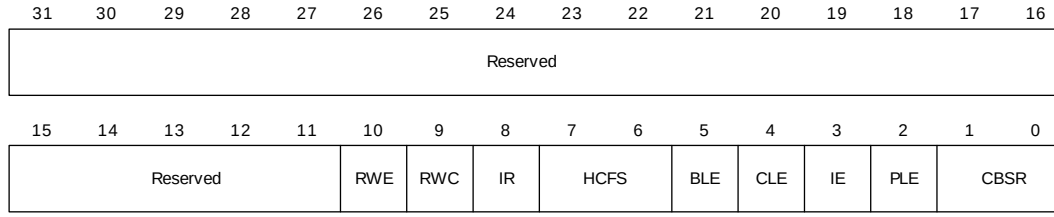
Bits	Access	Mnemonic	Reset	Description
D31:08	N/A	Reserved	N/A	N/A
D07:00	R	REV	10h	Version of the OHCI specification being used.

Table 455: HcRevision register

HcControl register

Address: 9080 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
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.

Table 456: HcControl register

Bits	Access	Mnemonic	Reset	Description
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.
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.

Table 456: HcControl register

Bits	Access	Mnemonic	Reset	Description
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.
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.

Table 456: HcControl register

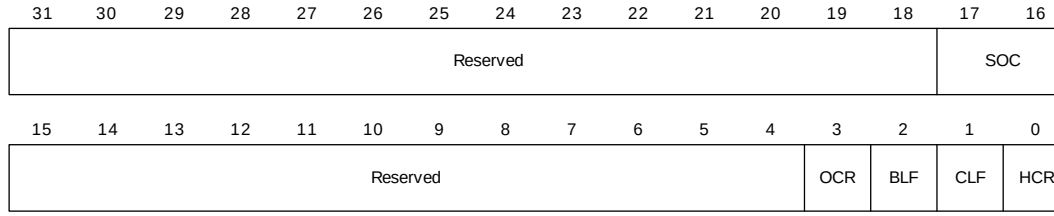
Bits	Access	Mnemonic	Reset	Description
D01:00	R/W	CBSR	00b	**ControlBulkServiceRatio** Specifies the service ratio between control and bulk endpoint descriptors (EDs). Before processing any of the nonperiodic lists, the Host controller must compare the ratio specified with its internal count on how many 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 456: HcControl register

HcCommandStatus register

Address: 9080 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.



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. 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 457: 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. 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 457: 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 457: HcCommandStatus register

HcInterruptStatus register

Address: 9080 100C

The HcInterruptStatus register provides status on various events that cause hardware interrupts. When an event occurs, the NS9360 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 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.

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. 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 or any HcRhPortStatus[NumberofDownstreamPort] register has changed.
D05	R/W	FNO	0b	**FrameNumberOverflow** Set when the most significant bit (MSB), bit 15, of the HcFmNumber 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.
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.

Table 458: HcInterruptStatus register

Bits	Access	Mnemonic	Reset	Description
D01	R/W	WDH	0b	**WritebackDoneHead** Set immediately after the Host controller has written HcDoneHead 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 to be incremented.

Table 458: HcInterruptStatus register

HcInterruptEnable register

Address: 9080 1010

Each enable bit in the HcInterruptEnable register corresponds to an associated interrupt bit in the HcInterrupt Status register. 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.

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.
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.

Table 459: HcInterruptEnable register

Bits	Access	Mnemonic	Reset	Description
D00	R/W	SO	0b	**Scheduling overrun** 0 Ignore 1 Enable interrupt generation due to scheduling overrun.

Table 459: HcInterruptEnable register

HcInterruptDisable register

Address: 9080 1014

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register works in conjunction with the HcInterruptEnable register. 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.

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.

Table 460: HcInterruptDisable register

Bits	Access	Mnemonic	Reset	Description
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.
D00	R/W	SO	0b	**Scheduling overrun** 0 Ignore 1 Disable interrupt generation due to scheduling overrun.

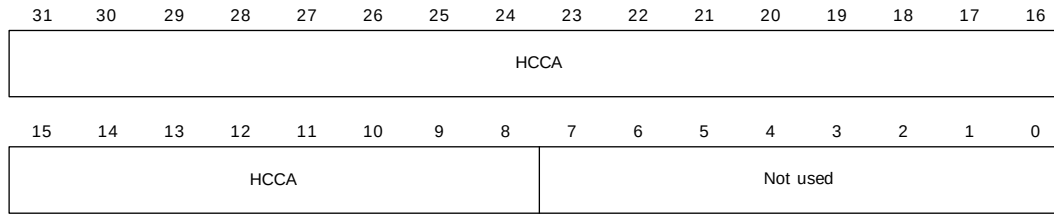
Table 460: HcInterruptDisable register

HcHCCA register

Address: 9080 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

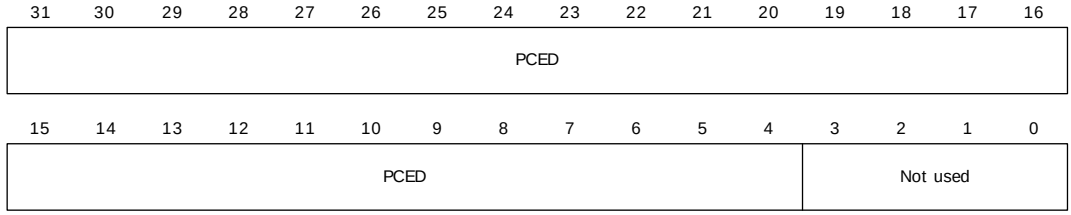
Bits	Access	Mnemonic	Reset	Description
D31:08	R/W	HCCA	0h	Base address of the Host controller communication area.
D07:00	R/W	Not used	0	Must be written to 0.

Table 461: HcHCCA register

HcPeriodCurrentED register

Address: 9080 101C

The HcPeriodCurrentED register contains the physical address of the current isochronous or interrupt endpoint descriptor.



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 462: HcPeriodCurrentED

HcControlHeadED register

Address: 9080 1020

The HcHeadControlED register contains the physical address of the first endpoint descriptor of the control list.

Register bit assignment

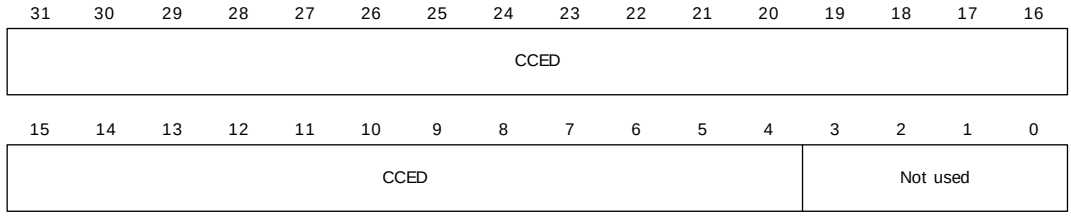
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	CHED	0h	**ControlHeadED** The Host controller traverses the control list starting with the HcCOntrolHeadED pointer. The content is loaded from the Host controller communication area during the Host controller initialization.
D03:00	N/A	Not used	0	Must be written to 0.

Table 463: HcControlHeadED register

HcControlCurrentED register

Address: 9080 1024

The HcControlCurrentED register contains the physical address of the control list's current endpoint descriptor.



Register bit assignment

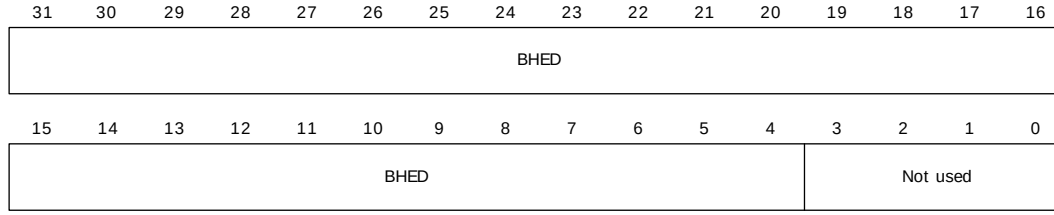
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	CCED	0h	**ControlCurrentED** This pointer is advanced to the next endpoint descriptor after serving the present one. The Host controller continues processing the list from where it left off in the last frame. When it reaches the end of the control list, the Host controller checks the ControlListFilled field. 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 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 464: HcControlCurrentED register

HcBulkHeadED register

Address: 9080 1028

The HcBulkHeadED register contains the physical address of the first endpoint descriptor of the bulk list.



Register bit assignment

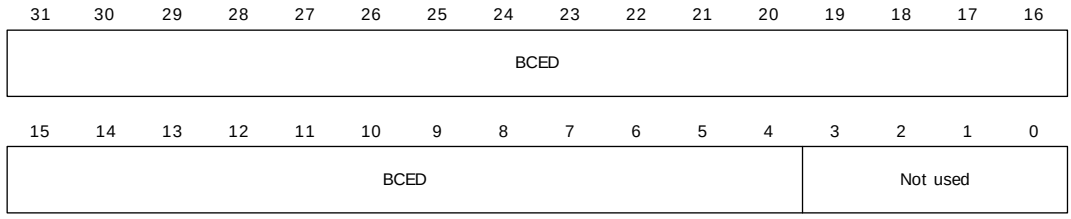
Bits	Access	Mnemonic	Reset	Description
D31:04	R/W	BHED	0h	BulkHeadED The Host controller traverses the bulk list starting with the HcBulkHeadED pointer. The content is loaded from the Host controller communication area during the Host controller initialization.
D03:00	N/A	Not used	0	Must be written to 0.

Table 465: HcBulkHeadED register

HcBulkCurrentED register

Address: 9080 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. 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 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 466: HcBulkCurrentED register

HcDoneHead register

Address: 9080 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

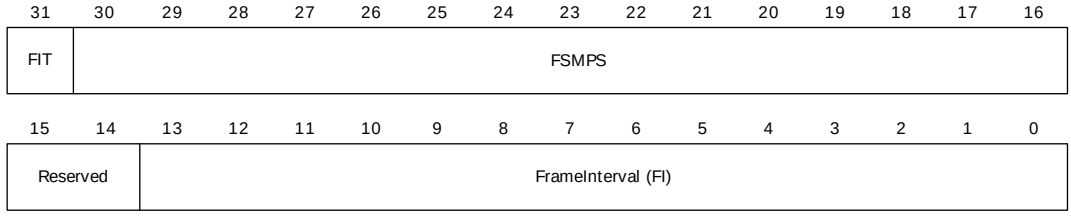
Bits	Access	Mnemonic	Reset	Description
D31:04	R	DH	0h	**DoneHead** When a TD is completed, the Host controller writes the content of HcDoneHead to the NextTD field of the TD. The Host controller then overwrites the content of HcDoneHead with the address of this TD. This value is written to zero whenever the Host controller writes the content of this register to the Host controller communications area. It also sets the WritebackDoneHead of the HcInterruptStatus register.
D03:00	N/A	Not used	0	Must be written to 0.

Table 467: HcDoneHead register

HcFmInterval register

Address: 9080 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 468: HcFmInterval register

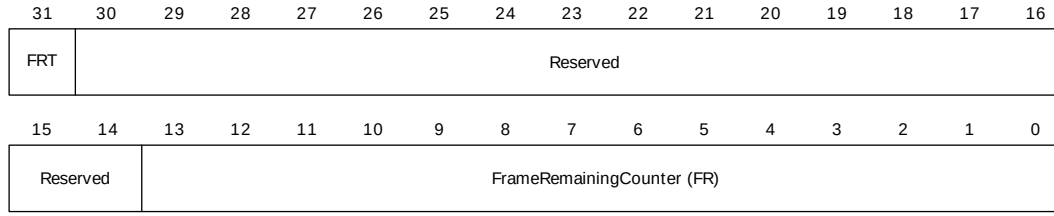
Bits	Access	Mnemonic	Reset	Description
D13:00	R/W	FI	2EDFh	**FrameInterval** Specifies the interval between two consecutive SOFs in bit times. The nominal value is 11,999. The Host controller driver should store the current value of this field before resetting the Host controller using the HostControllerReset field in the HcCommandStatus register, 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 468: HcFmInterval register

HcFmRemaining register

Address: 9080 1038

The HcFmRemaining register is a 14-bit down counter showing the bit time remaining in the current frame.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R	FRT	0b	**FrameRemainingToggle** Loaded from the FrameIntervalToggle field of the HcFmInterval register 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

Table 469: HcFmRemaining register

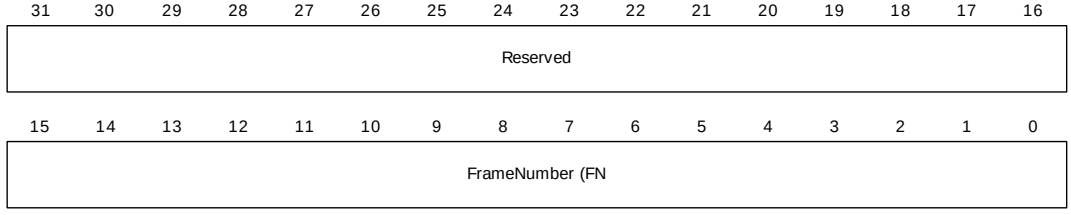
Bits	Access	Mnemonic	Reset	Description
D13:00	R	FR	0h	**FrameRemaining counter** Decremented at each bit time. When the counter reaches zero, it is reset by loading the FrameInterval value specified in the HcFrameInterval register at the next bit time boundary.

Table 469: HcFmRemaining register

HcFmNumber register

Address: 9080 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.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	N/A	Reserved	N/A	N/A

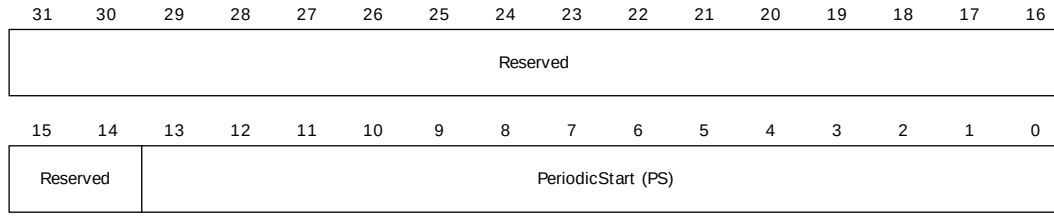
Table 470: HcFmNumber register

Bits	Access	Mnemonic	Reset	Description
D15:00	R	FN	0h	**FrameNumber** Incremented when the HcFmRemaining register 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.

Table 470: HcFmNumber register

HcPeriodicStart register

Address: 9080 1040



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:14	N/A	Reserved	N/A	N/A

Table 471: HcPeriodicStart register

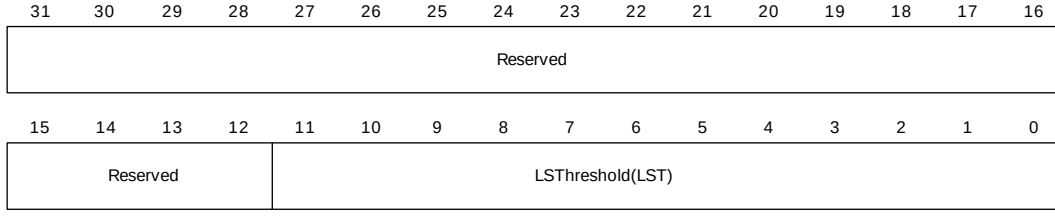
Bits	Access	Mnemonic	Reset	Description
D13:00	R/W	PS	0h	**PeriodicStart** Determines when is the earliest time the Host controller should start processing the periodic list. After a hardware reset, the PS field is cleared. The field is then set by the Host controller driver during the Host controller 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 471: HcPeriodicStart register

HcLSThreshold register

Address: 9080 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.



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 472: 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.

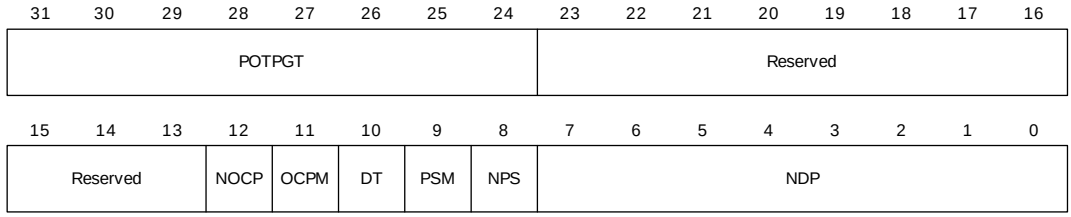
- The HcRhDescriptor A and HcRhDescriptorB registers should be implemented to be writeable, regardless of the Host controller USB state.
- The HcRhStatus register and HcRhPortStatus register must be writeable during the USB operational state.

Note: *IS* denotes an *implementation-specific* reset value for the related field.

HcRhDescriptorA register

Address: 9080 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.



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 port 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 473: HcRhDescriptorA register

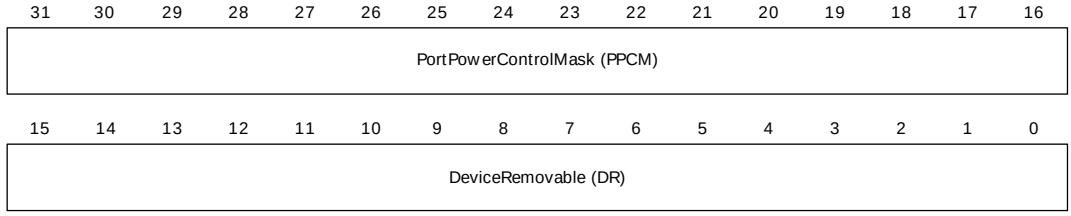
Bits	Access	Mnemonic	Reset	Description
D11	R/W	OCPM	IS	**OverCurrentProtectionMode** Describes how the overcurrent status for the root hub ports is reported. At reset, this field should reflect the same mode as PowerSwitchingMode (D08 in this register). The OCPM field is valid only if the NoOverCurrentProtection 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 hub 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 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 473: HcRhDescriptorA register

HcRhDescriptorB register

Address: 9080 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.



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 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

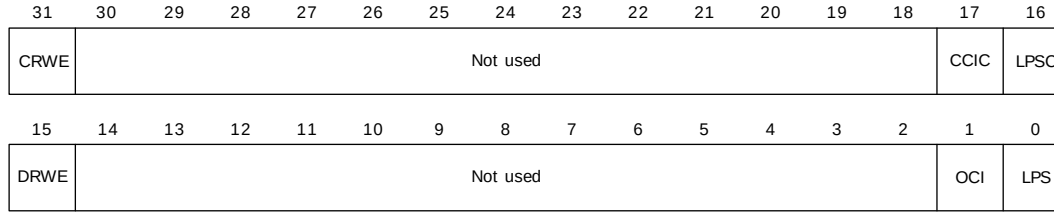
Bits	Access	Mnemonic	Reset	Description
D15:00	R/W	DR	IS	**DeviceRemovable** Each bit is dedicated to a root hub port. ■ When cleared, the attached device is removable. ■ When set, the attached device is not removable. Bit 0: Reserved Bit 1: Device attached to port #1 Bit 2: Device attached to port #2 ... Bit 15: device attached to port #15

HcRhStatus register

Address: 9080 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.



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 475: 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 475: HcRhStatus register

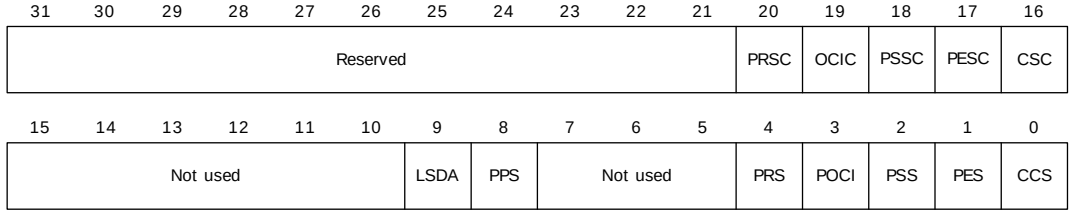
Bits	Access	Mnemonic	Reset	Description
D00	R/W	LPS	0b	**LocalPowerStatus (read)** Not supported; always read as 0. **ClearGlobalPower (write)** In global power mode (PowerSwitchingMode=0), this bit is always written to 1 to turn off power to all ports (clear PowerPortStatus). In per-port power mode, this bit clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing 0 to this bit has no effect.

Table 475: HcRhStatus register

HcRhPortStatus[1] register

Address: 9080 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.



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 the root hub changes the PortOverCurrentIndicator bit. The Host controller driver writes a 1 to clear this bit. Writing 0 has no effect.

Table 476: 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 476: 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 if 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 476: 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 476: 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 476: 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 NS9360 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 476: HcRhPortStatus[1] register

USB Device Module

CHAPTER 17

The USB Device module provides a USB 2.0 compliant interface. The USB Device module in the NS9360 supports both full-speed (12Mbps) and low-speed (1.5 Mbps) operation.

Note: You should be familiar with the NS9360 BBus DMA Controller design and the USB 2.0 specification before using full-speed and low-speed operation with the NS9360.

Overview

The USB Device module provides a USB 2.0 compliant interface for both full-speed (12Mbps) and low-speed (1.5 Mbps) operation. The module supports one bidirectional endpoint and up to 10 unidirectional endpoints that can be individually programmed for endpoint type (interrupt, bulk, or isochronous) and direction. Each endpoint is assigned to a DMA channel in a multi-channel BBus DMA controller. The USB Device module interfaces to either the internal NS9360 USB PHY or an external USB PHY using GPIO.

USB Device module architecture

The USB Device module consists of a USB Device controller, a multi-channel BBus DMA controller, and a USB Device Front End (UDFE) block that interfaces the USB Device controller to the DMA controller. The UDFE also provides several control and status registers for managing the USB Device module.

Figure 105 shows the USB Device module architecture.

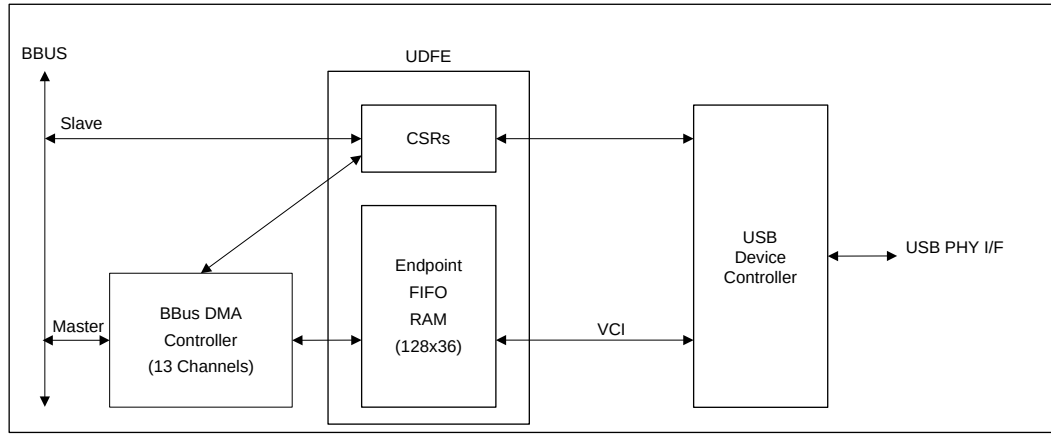


Figure 105: USB Device module architecture

The USB Device controller responds to packets initiated by the USB Host and is responsible for handling all details of the USB protocol (for example, bit-stuffing, CRC generation and checking, handshake generation, etc.). The controller is configured to

support up to 11 logical endpoints (one control endpoint and up to 10 unidirectional endpoints), and interfaces to the UDFE using a Virtual Component Interface (VCI). Some USB Commands (such as SET_CONFIGURATION, SET_INTERFACE, SET_ADDRESS) are processed by the USB Device controller without being passed onto the system. All other control and data packets are passed to system memory for software processing.

The UDFE and DMA controller provide the path for transferring the USB packets to and from system memory. The data to and from the USB Device controller is treated as a stream of bytes, and the UDFE provides no additional packet processing. Each of the endpoints has a separate 32-byte FIFO, which is implemented as eight locations in an embedded 128x36 dual-ported RAM.

Control endpoint #0 is the bidirectional endpoint — required by all USB Devices — and consists of two FIFOs:

- FIFO #1 handles USB-OUT packets for control endpoint #0.
- FIFO #2 handles USB-IN packets for control endpoint #0.

Endpoint characteristics (such as direction, type, and max size) are set in the related FIFO Status and Control registers (see page 750), FIFO Packet Control registers (see page 749), and Physical Endpoint Descriptor registers (see page 734).

USB Device controller features

The USB Device controller provides these features and capabilities:

- Supports USB 2.0 full-speed (12 Mbps) operation
- Supports USB 2.0 low-speed (1.5 Mbps) operation
- Supports up to 11 logical endpoints
- Clock and data recovery from USB
- Performs all CRC checking and generation
- Maintains data toggle bits
- Provides decoding of standard USB commands to control endpoint #0
- Supports up to 6 configurations, with each configuration supporting up to 5 interfaces and each interface handling up to 5 alternate settings

  Note: Only five configurations are supported if dynamic programming is enabled (CSRPRG bit set in the UDFE USB Device Controller Programming Control/Status register).

- Supports class/vendor commands by passing the SETUP transactions to the application
- Includes suspend/resume logic

USB-IN packet flow

The direction of data flow for USB-IN packets is from the USB Device to the USB Host. The DMA controller writes the data from system memory to the FIFO; the UDFE reads the data from the FIFO and sends it to the USB Device controller. The DMA controller continues to write data into the FIFO until it is full or has reached the end of the DMA transmit buffer. The USB Device controller always wants to transfer the maximum packet size of data, as indicated in the MAX field in the associated UDFE FIFO Packet Control register and the MAXSIZE field in the associated USB Device controller Physical Endpoint Descriptor register. The UDFE must tell the USB Device controller whether it prematurely runs out of data, so the USB Device controller will know when a transaction ends with less than a maximum packet size of data.

The UDFE returns a NAK to the USB Device controller if the FIFO is empty when the USB-IN packet arrives. In this situation, the NAK bit for the endpoint is set in the associated FIFO Interrupt Status register. In response to this interrupt, the software needs to set up a DMA transaction to load the FIFO. Because the USB Host will try this request again, subsequent USB-IN packets to this endpoint will also be NAKed until data is available in the FIFO.

USB-IN packet error handling

With USB-IN packets, the USB Host does not respond at all to indicate that there was an error in transmission. Therefore, the ERROR bit for the endpoint is set in the associated FIFO Interrupt Status register. The UDFE disables the endpoint's FIFO and flushes the contents by setting the CLR bit in the associated FIFO Status and Control register. To respond to the interrupt, you must perform these steps to retransmit the errored packet when the USB Host requests the next USB-IN packet from this endpoint:

1. Use the COUNT value in the associated FIFO Packet Control register to compute a new transmit buffer descriptor source address and buffer length, and update the current buffer descriptor in system memory. The current DMA buffer

descriptor can be determined by reading the index field in the channel's DMA Control register.

2 Write a 1 to the BDR and CE bits in the appropriate DMA Control register to force a refetch of the buffer descriptor when the channel next wins arbitration.

3 Take the USB Device endpoint FIFO out of reset by writing a 0 the CLR bit in the associated FIFO Status and Control register. At this point, the DMA controller refetches the buffer descriptor and the USB transfer restarts from the beginning if the errored packet.

USB-OUT packet flow

With USB-OUT packets, the direction of data flow is from the USB Host to the USB Device. The UDFE writes the packet data from the USB Device controller into the FIFO for the addressed endpoint, and the DMA controller reads the data from the FIFO and writes it into system memory. The DMA controller reads data from the FIFO only when these conditions exist:

- At least 16 bytes of data in the FIFO
- UDFE has identified the end of the USB transfer (that is, single or multi-packet data)
- UDFE has found a transmission error (see "USB-OUT packet error handling" on page 716)

At the end of the transfer, or when a transmission error is found, the UDFE tells the DMA controller to close its receive buffer for this endpoint.

The end of the USB-OUT transfer is indicated by one of these conditions:

- Receipt of an USB-OUT packet that is less than the maximum packet size, as indicated by the MAX field in the appropriate UDFE FIFO Packet Control register and the MAXSIZE field in the appropriate USB Device controller Physical Endpoint Descriptor register.
- Receipt of an USB-OUT packet of maximum packet size followed by a programmable period of inactivity defined by the TIMEOUT_CNT field in the associated FIFO Status and Control register.

TIMEOUT_CNT

If the TIMEOUT_CNT is greater than 0x0, a hardware timer is loaded with the TIMEOUT_CNT value each time data is written to the FIFO. The timer is decremented each time a USB-SOF packet (USB start-of-frame packet, which occurs at 1 msec rate) is found. If the timer decrements to 0, the UDFE tells the DMA controller to close its receive buffer for this endpoint because the FIFO has not been updated within a certain number of SOFs.

Any transfers that experience a timeout cause the TIMEOUT bit to be set in the associated FIFO Status and Control register. The buffer descriptor status for the transfer will also have the TIMEOUT bit set.

To minimize latency in processing control packets, do not use the TIMEOUT_CNT feature for the Control-OUT endpoint (that is, FIFO #1). Set the maximum packet size for the Control-OUT endpoint to 65, so all control packets are less than the maximum packet size of 64 bytes.

- Set the MAX field in FIFO Packet Control register #1 to 0x41.

  Note: The maximum packet size for the Control-IN endpoint remains set to 64 bytes.

- Set the MAXSIZE Field in the associated Physical Endpoint Descriptor register to 0x40.

Note: The TIMEOUT_CNT feature cannot be used for USB low-speed application because USB-SOF packets are not defined for USB low-speed. Since all packets are 8 bytes for USB low-speed applications, set the MAX field in *only* the appropriate FIFO Packet Control register to 9. All packets will then cause a receive buffer to be closed.

The MAXSIZE field in the associated Physical Endpoint Descriptor register must still be set to 0x8.

USB-OUT packet error handling

Both the USB Device controller and the UDFE can generate errors for USB-OUT packets.

USB Device controller error handling

The USB Device responds either with an ACK handshake to indicate a successful transmission or does not respond at all to indicate an error in transmission. The USB Device controller finds USB protocol errors (for example, CRC errors) and sends an indication of an unsuccessful packet to the UDFE. The UDFE then tells the DMA controller to close the receive buffer, and indicates that the packet experienced an error by clearing the M31 bit in the buffer descriptor status sent to the DMA controller. Software uses the status information to discard the last packet in the buffer. The USB Host will retransmit the errored packet and the DMA controller uses the next buffer descriptor to store it in system memory.

UDFE error handling

If a write is attempted to a full FIFO, the UDFE drops the rest of the packet and tells the USB Device not to send an ACK handshake for the packet. The UDFE then tells the DMA controller to close the receive buffer, and indicates that the packet experienced an error by clearing the M31 bit and setting the OVERFLOW bit in the buffer descriptor status sent to the DMA controller. Software uses the status information to discard the last packet in the buffer. The USB Host will retransmit the errored packet and the DMA controller uses the next buffer descriptor to store it in system memory. Once any portion of a packet has been dropped due to a full FIFO, the rest of the packet will be dropped even if the FIFO full condition is cleared.

When the UDFE requests that the DMA controller close the buffer, subsequent packets are dropped until the DMA controller acknowledges that the buffer has been closed. If the acknowledgement is received during transmission of another packet, that packet is dropped and the UDFE will start accepting packets at the end of the packet in progress.

USB STALL

The USB Device block supports issuing a USB STALL handshake in response to any USB-IN or USB-OUT packet on a per-endpoint basis. If the SEND_STALL bit is set in any of the FIFO Status and Control registers, a USB STALL handshake is sent as the response to any packet to the associated endpoint. If a USB STALL handshake is sent, the STALL_SENT bit in the associated FIFO Status Control register is set.

Control and Status registers

The Control and Status registers in the USB Device controller, DMA controller, and UDFE are accessed through the BBus slave interface. Only single, 32-bit transfers are supported. No burst accesses to these registers are allowed.

Logical and physical endpoints

The USB Device controller supports up to 11 physical endpoints: the bidirectional control endpoint #0 and 10 additional endpoints. The configuration, interface, and alternate settings for each endpoint are programmed using the Physical Endpoint Descriptor #0–#11 registers. 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 equal to the total number of logical endpoints in each alternate of each interface of each configuration. For example, if endpoint #3 has two different alternate settings for the same interface and configuration, it requires two physical endpoints.

DMA buffer descriptor status

The DMA buffer descriptor status field is updated when the buffer descriptor is retired. Table 477 defines the status bit fields for the USB Device buffer descriptors. See the BBus DMA controller chapter for more information.

Bits	Mnemonic	Description
D15:14	Not used	Read back as 0.
D13	M31	Successful transfer status bit. See Table 500: "FIFO Status and Control registers" on page 750.
D12	M30	Setup command status bit. See Table 500: "FIFO Status and Control registers" on page 750.
D11	OVFLOW	**Overflow field** Valid only for OUT direction endpoints. This bit is set if the endpoint's FIFO reaches a full state at any point during a transfer. This field reads back a 0 for IN direction endpoints.

Table 477: Status bit field definition

Bits	Mnemonic	Description
D10	TIMEOUT	**Timeout field** Valid only for OUT direction endpoints.This bit is set if the buffer is closed due to a timeout event. This field reads back a 0 for IN direction endpoints.
D09:00	Not used	Read back as zero.

Table 477: Status bit field definition

USB PHY options

The NS9360 supports interfacing the USB Device module to either the internal USB PHY provided by the NS9360 or an external USB PHY device. If a NS9360 application is using both the USB Host module and the USB Device module, the USB Device module must be connected to an external PHY device.

The NS9360 supports two types of external USB PHY devices: those with a bidirectional data interface and those with a unidirectional data interface. Table 478 shows how the NS9360 selects the different PHY options. The EXTPHY and INTPHY bits are discussed in the USB Configuration register in the BBus Utility chapter.

INTPHY	EXTPHY	USB PHY device option
1	X	Internal
0	0	External, bidirectional data
0	1	External, unidirectional data

Table 478: USB Device PHY selection

Figure 106 shows the NS9360 connected to a unidirectional USB PHY. Note that *RP* and *RM* are added to pull the data lines to the USB idle state during powerup and reset.

Notes:

1. All USB pins shown in NS9360 are GPIO. See the GPIO MUX pinout for pin assignments.
2. V+ is 3.3V for full speed and GND for low speed.
3. V- is GND for full speed and 3.3V for low speed.

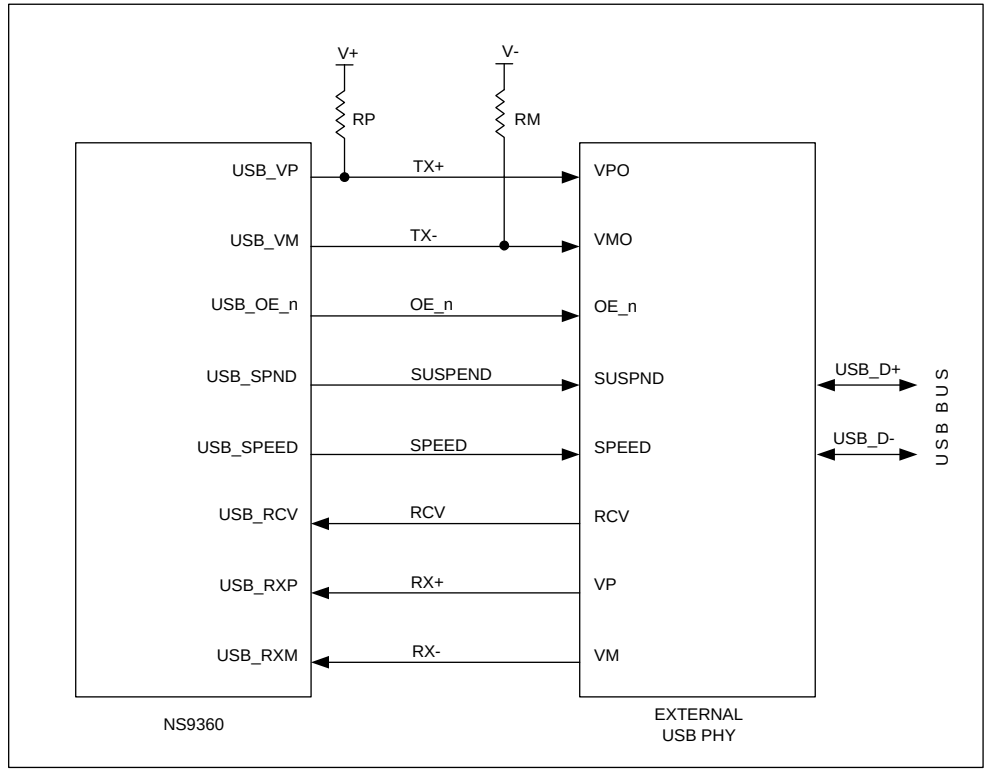


Figure 106: NS9360 connections to unidirectional external USB PHY

Figure 107 shows the NS9360 connected to an external bidirectional USB PHY.

- To prevent bus contention when both the USB PHY and the NS9360 are changing their data interfaces from input to output, OE_n is asserted before the NS9360 drives D+/D- as shown in Figure 108.
- To prevent bus contention when both the USB PHY and the NS9360 are changing their data interfaces from output to input, the NS9360 turns off D+/D- before OE_n is negated as shown in Figure 108.

To prevent glitching on the USB bus during these data transitions, RP and RM are added to pull the data lines to the USB idle state when D+/D- are floating.

Notes:

1 All USB pins shown in NS9360 are GPIO. See the GPIO MUX pinout for pin assignments.

2 V+ is 3.3V for full speed and GND for low speed.

3 V- is GND for full speed and 3.3V for low speed.

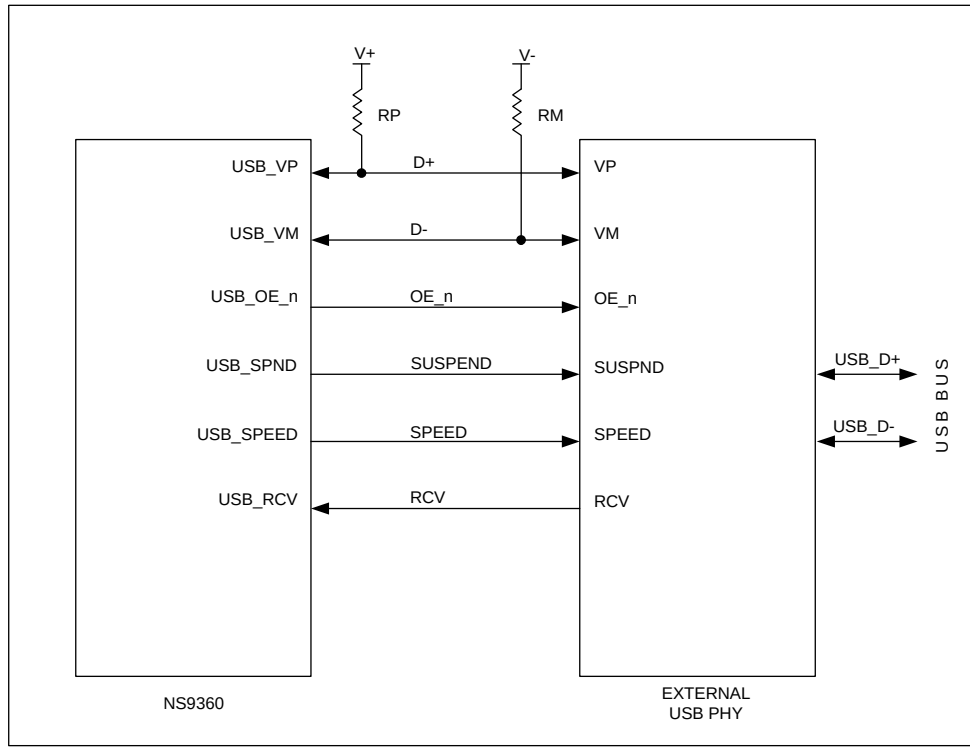


Figure 107: NS9360 connections to bidirectional external USB PHY

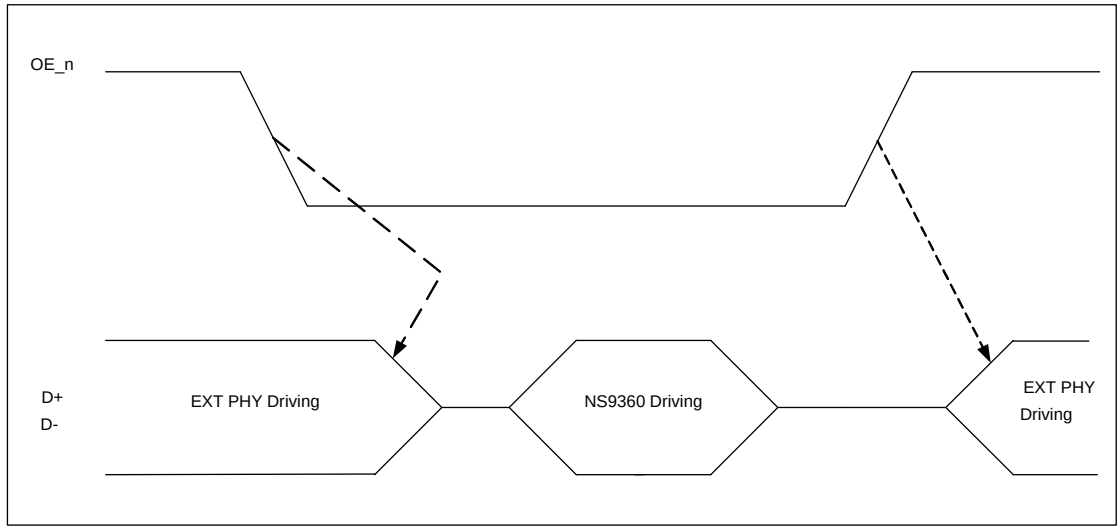


Figure 108: NS9360 data interface to bidirectional external USB PHY

System considerations

If the NS9360 is configured as a Big Endian system, the external inverter is required on the OE_n signal to an external USB PHY connected to gpio[49:42]. Figure 109 shows the inverter as U1.

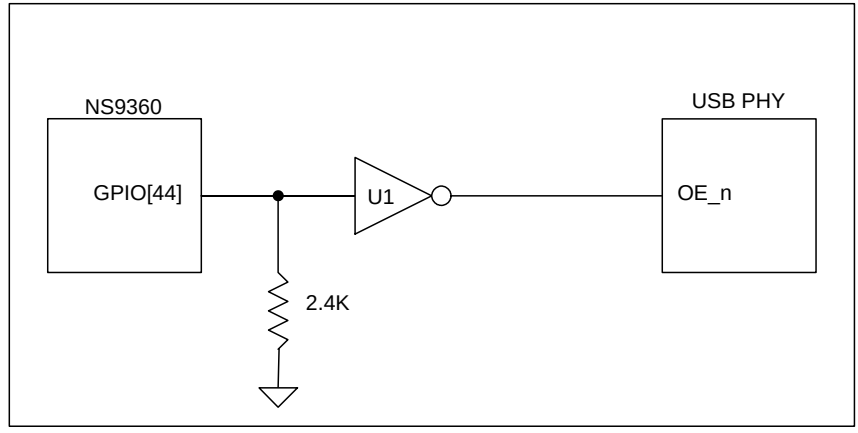


Figure 109: USB external PHY in a Big Endian system

Because gpio[44] must be pulled down low during powerup to initialize the NS9360 for Big Endian mode, the external inverter is used to guarantee that OE_n to the external PHY is held in the inactive state during this time. This allows the system to control when the NS9360 USB Device connects to the USB bus; otherwise, the USB Device is connected immediately after the NS9360 is powered up. In addition to the inverter, gpio[44] must also be configured, using GPIO Configuration Register #6 as an inverting output.

Note: This modification is not required for Little Endian systems.

USB Device module registers

The USB Device module configuration registers are located at base address 0x9090_0000. The next table provides the address register map for the USB Device module.

Address range	Register space (module)
0x90900000–0x90900FFF	UDFE Control and Status
0x90902000–0x90902FFF	USB Device Controller Block
0x90903000–0x90303FFF	UDFE Endpoint FIFOs
0x90910000–0x9091FFFF	USB Device DMA (see the BBus DMA Controller chapter for details about these registers)

Table 479: USB Device module register address map

UDFE registers

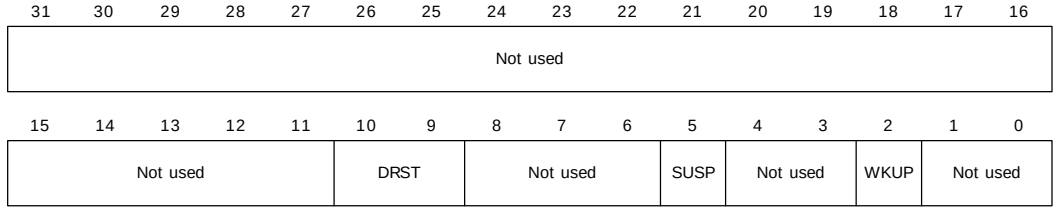
The next table provides the addresses for the UDFE registers. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register
9090 0000	UDFE Control/Status Register 0
9090 0004	UDFE Control/Status Register 1
9090 000C	UDFE Interrupt Enable
9090 0010	UDFE Interrupt Status
9090 0014	UDFE USB Device Controller Programming Control/Status

Table 480: USB Global registers address map

UDFE Control and Status Register 0

Address: 9090 0000



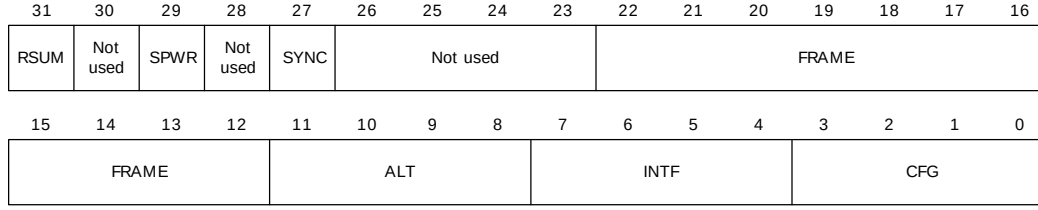
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:11	R	Not used	0x000000	Always read as 0.
D10:09	R	DRST	11	**Device reset** 00 Indicates that the device block is operational 11 Indicates that the device block is in reset Provides the device block reset status.
D08	R	Not used	0	Read only; any value.
D07:06	R	Not used	00	Always read as 0.
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	R	Not used	0	Always read as 0.
D03	R	Not used	0	Read only; any value.
D02	R/W	WKUP	0	**Wakeup** 0 Disable remote wakeup feature 1 Enable remote wakeup feature
D01:00	R	Not used	1	Always read as 0.

Table 481: UDFE Control and Status Register 0

UDFE Control and Status Register 1

Address: 9090 0004



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	R/W	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	R	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 NS9360 is always self-powered.
D28	R/W	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, which is used only by isochronous endpoints. 0 SYNC_FRAME packet is not supported 1 SYNC_FRAME packet is supported When set to 0, the USB Device replies to a SYNC_FRAME packet with a STALL handshake.
D26:23	R	Not used	0	Read only; any value.
D22:12	R	FRAME	0x000	**Frame number** Contains the current frame number. **Note:** This field is used for diagnostic purposes only.

Table 482: UDFE Control and Status Register 1

Bits	Access	Mnemonic	Reset	Description
D11:08	R	ALT	0x0	**Alternate value** Contains the current alternate value for the device block. **Note:** This field is used for diagnostic purposes only.
D07:04	R	INTF	0x0	**Interface value** Contains the current interface value for the device block. **Note:** This field is used for diagnostic purposes only.
D03:00	R	CFG	0x0	**Configuration value** Contains the current configuration value for the device block. **Note:** This field is used for diagnostic purposes only.

Table 482: UDFE Control and Status Register 1

UDFE Interrupt Enable register

Address: 9090 000C

The UDFE Interrupt Enable register contains the 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	Not used	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	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	Not used	000	Always read as 0.

Table 483: UDFE Interrupt Enable register

Bits	Access	Mnemonic	Reset	Description
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	Not used	0	Always write to 0.
D25	R/W	DMA12	0	DMA channel 12 interrupt enable
D24	R/W	DMA11	0	DMA channel 11 interrupt enable
D23	R/W	DMA10	0	DMA channel 10 interrupt enable
D22	R/W	DMA9	0	DMA channel 9 interrupt enable
D21	R/W	DMA8	0	DMA channel 8 interrupt enable
D20	R/W	DMA7	0	DMA channel 7 interrupt enable
D19	R/W	DMA6	0	DMA channel 6 interrupt enable
D18	R/W	DMA5	0	DMA channel 5 interrupt enable
D17	R/W	DMA4	0	DMA channel 4 interrupt enable
D16	R/W	DMA3	0	DMA channel 3 interrupt enable
D15	R/W	DMA2	0	DMA channel 2 interrupt enable
D14	R/W	DMA1	0	DMA channel 1 interrupt enable
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 in the associated FIFO Interrupt Enable register.
D11	R/W	URST	0	USB bus reset interrupt enable
D10	R/W	SOF	0	SOF (start-of-frame) packet interrupt enable
D09	R/W	SSPND	0	SUSPEND interrupt enable
D08	R/W	SETINTF	0	SET INTERFACE packet interrupt enable
D07	R/W	SETCFG	0	SET CONFIGURATION packet interrupt enable
D06	R/W	Not used	0	Always write to 0
D05:00	R	Not used	0x00	Always read as 0.

Table 483: UDFE Interrupt Enable register

UDFE Interrupt Status register

Address: 9090 0010

The UDFE Interrupt Status register contains 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 Device BBus DMA controller block. The FIFO interrupts must be serviced in the appropriate FIFO Interrupt Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not used				GBL_ DMA	Not used	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	Rsvd	Not used					

Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:28	R	Not used	0	Always read as 00.
D27	R	GBL_DMA	0	Bit-wise logical OR of the DMA# fields (D26:14 in this register).
D26	R	Not used	0	N/A
D25	R	DMA12	0	DMA channel 12 interrupt. Service in the USB Device BBus DMA controller block.
D24	R	DMA11	0	DMA channel 11 interrupt. Service in the USB Device BBus DMA controller block.
D23	R	DMA10	0	DMA channel 10 interrupt. Service in the USB Device BBus DMA controller block.
D22	R	DMA9	0	DMA channel 9 interrupt. Service in the USB Device BBus DMA controller block.
D21	R	DMA8	0	DMA channel 8 interrupt. Service in the USB Device BBus DMA controller block.
D20	R	DMA7	0	DMA channel 7 interrupt. Service in the USB Device BBus Device DMA controller block.
D19	R	DMA6	0	DMA channel 6 interrupt. Service in the USB Device BBus DMA controller block.
D18	R	DMA5	0	DMA channel 5 interrupt. Service in the USB Device BBus DMA controller block.
D17	R	DMA4	0	DMA channel 4 interrupt. Service in the USB Device BBus DMA controller block.
D16	R	DMA3	0	DMA channel 3 interrupt. Service in the USB Device BBus DMA controller block.
D15	R	DMA2	0	DMA channel 2 interrupt. Service in the USB Device BBus DMA controller block.
D14	R	DMA1	0	DMA channel 1 interrupt. Service in the USB Device BBus DMA controller block.
D13	R	Not used	0	Always read as 0.

Table 484: UDFE Interrupt Status register

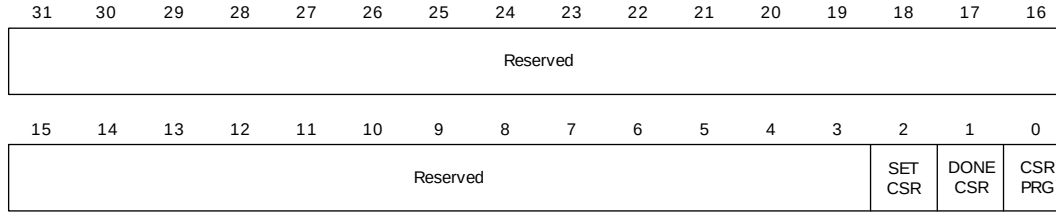
Bits	Access	Mnemonic	Reset	Description
D12	R	FIFO	0	Bit-wise logical OR of the enabled FIFO interrupt status fields in the FIFO Interrupt Status registers (see page 737).
D11	RW1TC	URST	0	Asserted when a USB reset is detected on the bus.
D10	RW1TC	SOF	0	Asserted when a SOF (start-of-frame) packet is received.
D09	RW1TC	SSPND	0	**SUSPEND** Asserted when the USB Device block has entered the SUSPEND state.
D08	RW1TC	SETINTF	0	**SET INTERFACE** Asserted when a SET INTERFACE packet is received.
D07	RW1TC	SETCFG	0	**SET CONFIGURATION** Asserted when a SET CONFIGURATION packet is received.
D06	N/A	Reserved	N/A	N/A
D05:00	R	Not used	0x00	Always read as 0.

Table 484: UDFE Interrupt Status register

UDFE USB Device Controller Programming Control/Status register

Address: 9090 0014

The USB Device Controller Programming Control/Status register contains the dynamic programming control and status information. The register allows you to reconfigure the USB Device controller when a SET_CONFIGURATION or SET_INTERFACE 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 in the USB Device controller. Must be set to 1 before programming can begin.
D01	R/W	DONECSR	0	**CSR programming done** Indicates to the USB Device controller that software has finished programming the CSRs. A value of 1 indicates that software is finished.
D00	R/W	CSRPRG	0	**CSR dynamic programming support** Enables dynamic programming support in the USB Device controller. Program this field on powerup, then leave it unchanged. Write a 1 to enable this feature.

Table 485: UDFE USB Device Controller Programming Control/Status register

USB Device controller registers

The next table provides the addresses of the USB Device controller registers.

Address	Register
9090 2000	Device Descriptor/Setup Command register
9090 2004	Physical Endpoint Descriptor #0
9090 2008	Physical Endpoint Descriptor #1
9090 200C	Physical Endpoint Descriptor #2
9090 2010	Physical Endpoint Descriptor #3

Table 486: USB Device controller registers address map

Address	Register
9090 2014	Physical Endpoint Descriptor #4
9090 2018	Physical Endpoint Descriptor #5
9090 201C	Physical Endpoint Descriptor #6
9090 2020	Physical Endpoint Descriptor #7
9090 2024	Physical Endpoint Descriptor #8
9090 2028	Physical Endpoint Descriptor #9
9090 202C	Physical Endpoint Descriptor #10
9090 2030	Physical Endpoint Descriptor #11

Table 486: USB Device controller registers address map

Device Descriptor/Setup Command register

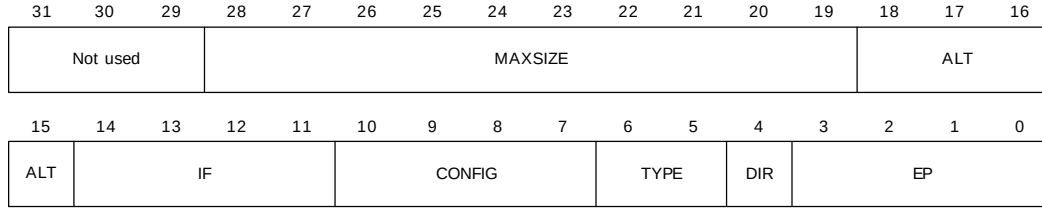
Address: 9090 2000

The Device Descriptor/Setup Command register is a legacy register. This register must be written to the value 0x0000_0100.

Physical Endpoint Descriptor #0–#11 registers

Address: 9090 2004 / 2008 / 200C / 2010 / 2014 / 2018 / 201C / 2020 / 2024 / 2028 / 202C / 2030

The Physical Endpoint Descriptor registers store the endpoint information. There are 12 registers, one for each endpoint descriptor. Each register contains the same information for each of the endpoint descriptors.



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:29	R/W	Not used	0x0	Always write to 0.
D28:19	R/W	MAXSIZE	10'h0	Maximum packet size. Set to the same value as the MAX field in the associated FIFO Packet Control register (see page 749).
D18:15	R/W	ALT	4'h0	Alternate setting to which this endpoint belongs. Only values 4'h0–4'h4 are supported.
D14:11	R/W	IF	4'h0	Interface number to which this endpoint belongs. Only values 4'h0–4'h4 are supported.
D10:07	R/W	CONFIG	4'h0	Configuration number to which this endpoint belongs. Only values 4'h0–4'h5 are supported. **Note:** Value of 0x0 is not supported if dynamic programming is enabled (CSRPRG set to 1 in the UDFE USB Device Controller Programming Control/Status register).
D06:05	R/W	TYPE	2'h0	**Endpoint type** 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Table 487: Physical Endpoint Descriptor register (for endpoint descriptors 0–11)

Bits	Access	Mnemonic	Reset	Description
D04	R/W	DIR	1h'0	**Endpoint direction** 0 Out 1 In
D03:00	R/W	EP	4'h0	**Endpoint number** Only values 4'h0–4'ha are supported.

Table 487: Physical Endpoint Descriptor register (for endpoint descriptors 0–11)

UDFE Endpoint FIFO Control registers

The next table provides the addresses for the UDFE Endpoint FIFO Control registers. All configuration registers must be accessed as 32-bit words and as single accesses only. Bursting is not allowed.

Address	Register
9090 3000	FIFO Interrupt Status 0
9090 3004	FIFO Interrupt Enable 0
9090 3010	FIFO Interrupt Status 1
9090 3014	FIFO Interrupt Enable 1
9090 3020	FIFO Interrupt Status 2
9090 3024	FIFO Interrupt Enable 2
9090 3030	FIFO Interrupt Status 3
9090 3034	FIFO Interrupt Enable 3
9090 3080	FIFO Packet Control #1
9090 3084	FIFO Packet Control #2
9090 3088	FIFO Packet Control #3
9090 308C	FIFO Packet Control #4
9090 3090	FIFO Packet Control #5

Table 488: UDFE Endpoint FIFO Control registers address map

Address	Register
9090 3094	FIFO Packet Control #6
9090 3098	FIFO Packet Control #7
9090 309C	FIFO Packet Control #8
9090 30A0	FIFO Packet Control #9
9090 30A4	FIFO Packet Control #10
9090 30A8	FIFO Packet Control #11
9090 30AC	FIFO Packet Control #12
9090 3100	FIFO Status and Control #1
9090 3108	FIFO Status and Control #2
9090 3110	FIFO Status and Control #3
9090 3118	FIFO Status and Control #4
9090 3120	FIFO Status and Control #5
9090 3128	FIFO Status and Control #6
9090 3130	FIFO Status and Control #7
9090 3138	FIFO Status and Control #8
9090 3140	FIFO Status and Control #9
9090 3148	FIFO Status and Control #10
9090 3150	FIFO Status and Control #11
9090 3158	FIFO Status and Control #12

Table 488: UDFE Endpoint FIFO Control registers address map

The next table 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

Table 489: FIFO to DMA channel to endpoint map

DMA channel	FIFO	EP number
4	4	2
5	5	3
6	6	4
7	7	5
8	8	6
9	9	7
10	10	8
11	11	9
12	12	10

Table 489: FIFO to DMA channel to endpoint map

UDFE FIFO Interrupt Status registers

The UDFE FIFO Interrupt Status registers contain interrupt status information for the UDFE endpoint 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 *write-1-to-clear* 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.

USB Device endpoint status

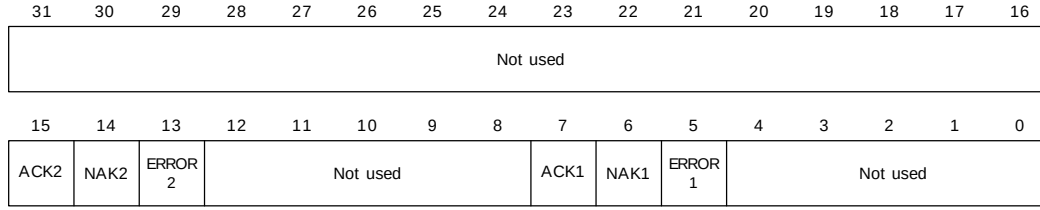
The next table defines the device endpoint status provided for each endpoint FIFO.

Status	Direction	Description
ACK	In	Set when an ACK handshake is received from the USB Host in response to a prior packet. For isochronous endpoints, this field is asserted automatically for each packet sent to the USB Host as no handshake is required.
NAK	In	Set when a NAK handshake is sent to the USB Host in response to the prior packet requested from the USB Host. This occurs when data is not yet available for this endpoint.
ERROR	In	Set when an ACK handshake is not received from the USB Host in response to the prior data packet sent to the USB Host. This bit is also set when a STALL response is sent because the STALL_SEND bit for this endpoint is set in the FIFO Interrupt Status and Control register for the addressed endpoint.
ACK	Out	Set when an ACK handshake is sent to the USB Host in response to the prior error-free packet received from the USB Host. For isochronous endpoints, this field is asserted if the data was received error-free as no handshake is required.
NAK	Out	Set when a NAK handshake is sent to the USB Host in response to the prior data packet received from the USB Host. This occurs when a packet is dropped because the FIFO for the addressed endpoint is full or the previous packet is still in the FIFO waiting for acknowledgement from the DMA controller.
ERROR	Out	Set when a packet is received from the USB Host and a transmission error was found. This bit is also set when a STALL response is sent because the STALL_SEND bit for this endpoint is set in the FIFO Interrupt Status and Control register for the addressed endpoint.

Table 490: USB Device endpoint status

FIFO Interrupt Status 0 register

Address: 9090 3000



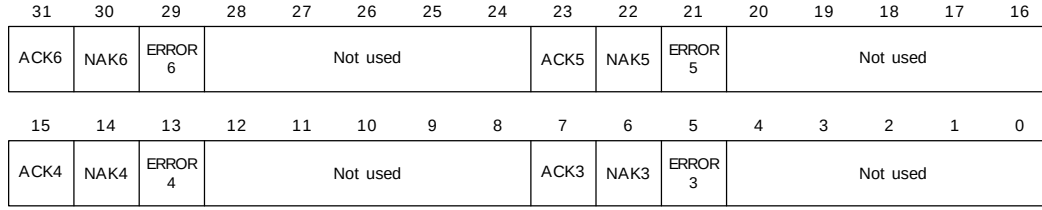
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R	Not used	0x0000	Always read as 0x0000.
D15	RW1TC	ACK2	0	Endpoint 0 (CTRL-In) acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D14	RW1TC	NAK2	0	Endpoint 0 (CTRL-In) negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D13	RW1TC	ERROR2	0	Endpoint 0 (CTRL-In) error status. See Table 490, “USB Device endpoint status,” on page 738.
D12:08	RW1TC	Not used	0x00	Always write to 00000.
D07	RW1TC	ACK1	0	Endpoint 0 (CTRL-Out) acknowledge status. See Table 490, “USB Device endpoint status,” on page 738
D06	RW1TC	NAK1	0	Endpoint 0 (CTRL-Out) negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D05	RW1TC	ERROR1	0	Endpoint 0 (CTRL-Out) error status. See Table 490, “USB Device endpoint status,” on page 738.
D04:00	RW1TC	Not used	0x00	Always write to 00000.

Table 491: FIFO Interrupt Status 0 register

FIFO Interrupt Status 1 register

Address: 9090 3010



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	ACK6	0	Endpoint 4 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D30	RW1TC	NAK6	0	Endpoint 4 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D29	RW1TC	ERROR6	0	Endpoint 4 error status. See Table 490, “USB Device endpoint status,” on page 738.
D28:24	RW1TC	Not used	0x00	Always write to 00000.
D23	RW1TC	ACK5	0	Endpoint 3 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D22	RW1TC	NAK5	0	Endpoint 3 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D21	RW1TC	ERROR5	0	Endpoint 3 error status. See Table 490, “USB Device endpoint status,” on page 738.
D20:16	RW1TC	Not used	0x00	Always write to 00000.
D15	RW1TC	ACK4	0	Endpoint 2 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D14	RW1TC	NAK4	0	Endpoint 2 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D13	RW1TC	ERROR4	0	Endpoint 2 error status. See Table 490, “USB Device endpoint status,” on page 738.
D12:08	RW1TC	Not used	0x00	Always write to 00000.
D07	RW1TC	ACK3	0	Endpoint 1 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.

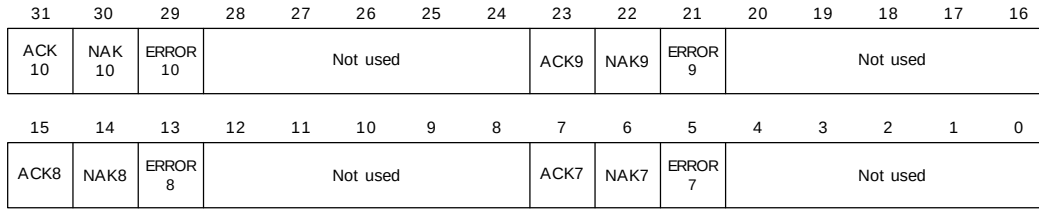
Table 492: FIFO Interrupt Status 1 register

Bits	Access	Mnemonic	Reset	Description
D06	RW1TC	NAK3	0	Endpoint 1 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D05	RW1TC	ERROR3	0	Endpoint 1 error status. See Table 490, “USB Device endpoint status,” on page 738.
D04:00	RW1TC	Not used	0x00	Always write to 00000.

Table 492: FIFO Interrupt Status 1 register

FIFO Interrupt Status 2 register

Address: 9090 3020



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31	RW1TC	ACK10	0	Endpoint 8 acknowledges status. See Table 490, “USB Device endpoint status,” on page 738.
D30	RW1TC	NAK10	0	Endpoint 8 negative acknowledgement status. See Table 490, “USB Device endpoint status,” on page 738.
D29	RW1TC	ERROR10	0	Endpoint 8 error status. See Table 490, “USB Device endpoint status,” on page 738.
D28:24	RW1TC	Not used	0x00	Always write to 00000.
D23	RW1TC	ACK9	0	Endpoint 7 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D22	RW1TC	NAK9	0	Endpoint 7 negative acknowledgement status. See Table 490, “USB Device endpoint status,” on page 738.
D21	RW1TC	ERROR9	0	Endpoint 7 error status. See Table 490, “USB Device endpoint status,” on page 738.

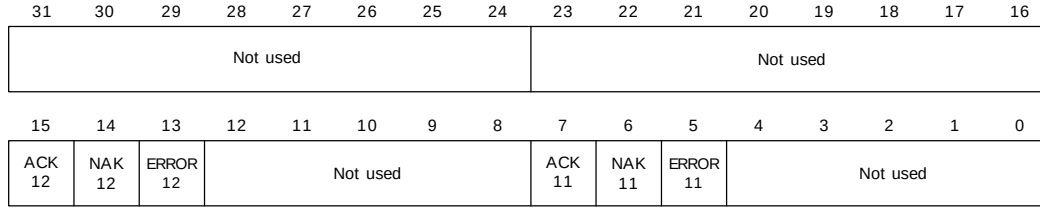
Table 493: FIFO Interrupt Status 2 register

Bits	Access	Mnemonic	Reset	Description
D20:16	RW1TC	Not used	0x00	Always write to 00000.
D15	RW1TC	ACK8	0	Endpoint 6 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D14	RW1TC	NAK8	0	Endpoint 6 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D13	RW1TC	ERROR8	0	Endpoint 6 error status. See Table 490, “USB Device endpoint status,” on page 738.
D12:08	RW1TC	Not used	0x00	Always write to 00000.
D07	RW1TC	ACK7	0	Endpoint 5 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D06	RW1TC	NAK7	0	Endpoint 5 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D05	RW1TC	ERROR7	0	Endpoint 5 error status. See Table 490, “USB Device endpoint status,” on page 738.
D04:00	RW1TC	Not used	0x00	Always write to 00000.

Table 493: FIFO Interrupt Status 2 register

FIFO Interrupt Status 3 register

Address: 9090 3030



Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:24	R	Not used	0x00	Always read as 0x00.
D23:16	RW1TC	Not used	0x00	Always write to 00000.
D15	RW1TC	ACK12	0	Endpoint 10 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D14	RW1TC	NAK12	0	Endpoint 10 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D13	RW1TC	ERROR12	0	Endpoint 10 error status. See Table 490, “USB Device endpoint status,” on page 738.
D12:08	RW1TC	Not used	0x00	Always write to 00000.
D07	RW1TC	ACK11	0	Endpoint 9 acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D06	RW1TC	NAK11	0	Endpoint 9 negative acknowledge status. See Table 490, “USB Device endpoint status,” on page 738.
D05	RW1TC	ERROR11	0	Endpoint 9 error status. See Table 490, “USB Device endpoint status,” on page 738.
D04:00	RW1TC	Not used	0x00	Always write to 00000.

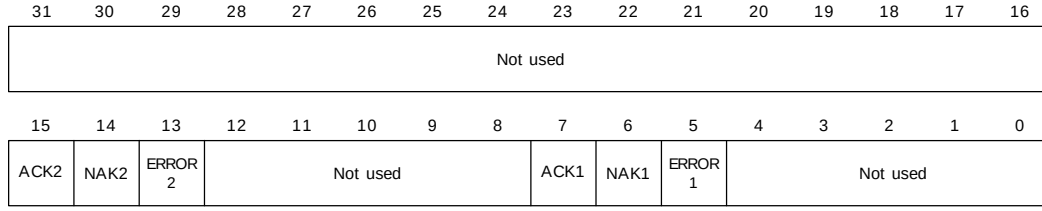
Table 494: FIFO Interrupt Status 3 register

FIFO Interrupt Enable registers

The FIFO Interrupt Enable registers contain the interrupt enable information for the UDFE Endpoint 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: 9090 3004



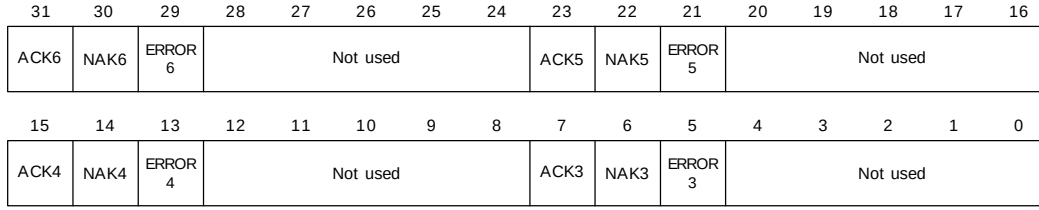
Register bit assignment

Bits	Access	Mnemonic	Reset	Description
D31:16	R	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	NAK2	0	Generate an interrupt when NAK2 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	R/W	Not used	0x00	Always write to 0x00000.
D07	R/W	ACK1	0	Generate an interrupt when ACK1 in FIFO Interrupt Status 0 register is asserted.
D06	R/W	NAK1	0	Generate an interrupt when NAK1 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	R/W	Not used	0x00	Always write to 0x00000.

Table 495: FIFO Interrupt Enable 0 register

FIFO Interrupt Enable 1 register

Address: 9090 3014



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	NAK6	0	Generate an interrupt when NAK6 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	R/W	Not used	0x00	Always write to 0x00000.
D23	R/W	ACK5	0	Generate an interrupt when ACK5 in FIFO Interrupt Status 1 register is asserted.
D22	R/W	NAK5	0	Generate an interrupt when NAK5 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	R/W	Not used	0x00	Always write to 0x00000.
D15	R/W	ACK4	0	Generate an interrupt when ACK4 in FIFO Interrupt Status 1 register is asserted.
D14	R/W	NAK4	0	Generate an interrupt when NAK4 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	R/W	Not used	0x00	Always write to 0x00000.

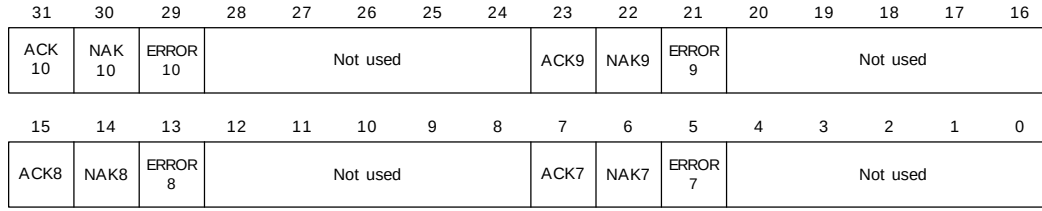
Table 496: FIFO Interrupt Enable 1 register

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	NAK3	0	Generate an interrupt when NAK3 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	R/W	Not used	0x00	Always write to 0x00000.

Table 496: FIFO Interrupt Enable 1 register

FIFO Interrupt Enable 2 register

Address: 9090 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	NAK10	0	Generate an interrupt when NAK10 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	R/W	Not used	0x00	Always write to 0x00000.
D23	R/W	ACK9	0	Generate an interrupt when ACK9 in FIFO Interrupt Status 2 register is asserted.
D22	R/W	NAK9	0	Generate an interrupt when NAK9 in FIFO Interrupt Status 2 register is asserted.

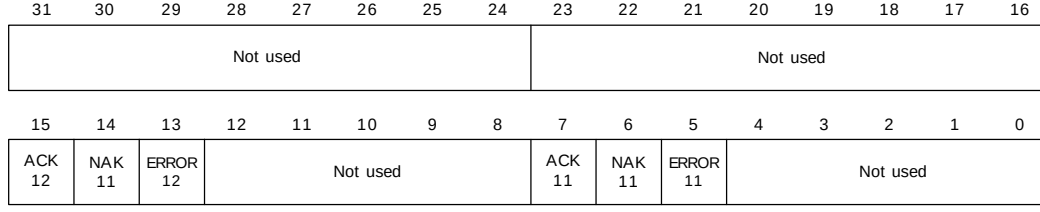
Table 497: FIFO Interrupt Enable 2 register

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	R/W	Not used	0x00	Always write to 0x00000.
D15	R/W	ACK8	0	Generate an interrupt when ACK8 in FIFO Interrupt Status 2 register is asserted.
D14	R/W	NAK8	0	Generate an interrupt when NAK8 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	R/W	Not used	0x00	Always write to 0x00000.
D07	R/W	ACK7	0	Generate an interrupt when ACK7 in FIFO Interrupt Status 2 register is asserted.
D06	R/W	NAK7	0	Generate an interrupt when NAK7 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	R/W	Not used	0x00	Always write to 0x00000.

Table 497: FIFO Interrupt Enable 2 register

FIFO Interrupt Enable 3 register

Address: 9090 3034



Register bit assignment

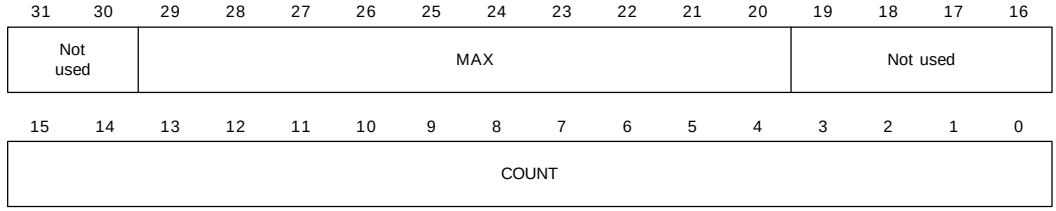
Bits	Access	Mnemonic	Reset	Description
D31:24	R	Not used	0x00	Always read as 0x00.
D23:16	R/W	Not used	0x00	Always write to 0x00000.
D15	R/W	ACK12	0	Generate an interrupt when ACK12 in FIFO Interrupt Status 3 register is asserted.
D14	R/W	NAK12	0	Generate an interrupt when NAK12 in FIFO Interrupt Status 3 register is asserted.
D13	R/W	ERROR12	0	Generate an interrupt when ERROR12 in FIFO Interrupt Status 3 register is asserted.
D12:08	R/W	Not used	0x00	Always write to 0x00000.
D07	R/W	ACK11	0	Generate an interrupt when ACK11 in FIFO Interrupt Status 3 register is asserted,
D06	R/W	NAK11	0	Generate an interrupt when NAK11 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	R/W	Not used	0x00	Always write to 0x00000.

Table 498: FIFO Interrupt Enable 3 register

FIFO Packet Control registers

Address: 9090 3080 / 3084 / 3088 / 308C / 3090 / 3094 / 3098 / 309C / 30A0 / 30A4 / 30A8 / 30AC

The FIFO Packet Control registers contain packet information for the UDFE Endpoint FIFOs. There are 12 of these registers, one for each non-control endpoint and the two required for the bidirectional control endpoint.



Register bit assignment

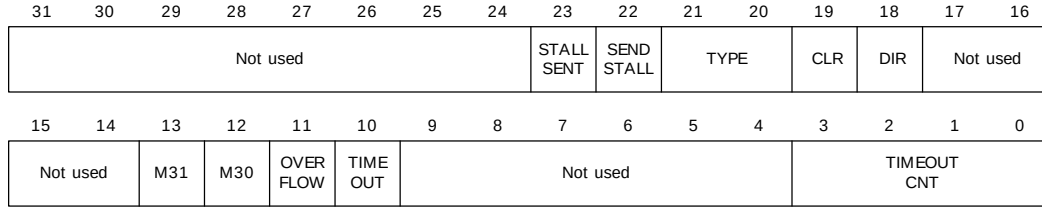
Bits	Access	Mnemonic	Reset	Description
D31:30	R	Not used	0x0	Always read as 0x0.
D29:20	R/W	MAX	0x040	Indicates the maximum packet size supported by the associated USB Device endpoint. This value should be set to the same value as the maximum packet size in the associated Physical Endpoint Descriptor register (see "Physical Endpoint Descriptor #0–#11 registers," beginning on page 734. **Note:** This field does not apply for FIFO #2 as that FIFO is dedicated to the IN direction of the control endpoint.
D19:16	R	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 499: FIFO Packet Control registers

FIFO Status and Control registers

Address: 9090 3100 / 3108 / 3110 / 3118 / 3120 / 3128 / 3130 / 3138 / 3140 / 3148 / 3150 / 3158

The FIFO Status and Control registers contain additional status and control information for the UDFE Endpoint FIFOs. There are 12 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	R	Not used	0x00	Always read as 0x00.
D23	RW1TC	STALL_SENT	0	**STALL_SENT** Set to 1 when the USB Device sends a STALL response to the USB Host when the SEND_STALL bit (D22) is set to 1. For diagnostics, you can set tis bit to 1 by writing a 1 when the bit is set to 0.
D22	R/W	SEND_STALL	0	**SEND_STALL** Tells the USB Device to send a STALL response to the USB bus for all IN and OUT packets when this bit is set to 1. The USB Device continues to send a STALL response until this bit is cleared.
D21:20	R/W	TYPE	0x0	**Type field** Defines the endpoint type associated with the FIFO. 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Table 500: FIFO Status and Control registers

Bits	Access	Mnemonic	Reset	Description
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.
D17:14	R	Not used	0x0	Always read as 0x0.
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 USB Host. ■ For OUT and SETUP packets: When set, this bit also indicates that an ACK handshake is being sent to the USB 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. ■ This bit is also cleared when a NAK or STALL handshake is being sent to the host.
D12	R	M30	0	**Setup command status bit** 0 Current transaction is not a setup command 1 Current transaction is a setup command

Table 500: FIFO Status and Control registers

Bits	Access	Mnemonic	Reset	Description
D11	R	OVERFLOW	0	**Overflow status bit** 0 No overflow detected during a USB-OUT packet to this FIFO. 1 FIFO overflowed during a USB-OUT packet to this FIFO.
D10	R	TIMEOUT	0	**Timeout status bit** 0 The DMA buffer receive for this USB-OUT packet was not closed because of a timeout condition. 1 The DMA receive buffer for this USB-OUT packet was closed because of a timeout condition.
D09:04	R	Not used	0x00	Always read as 0x00.
D03:00	R/W	TIMEOUT_CNT	0x000	**Timeout Count** Timeout count, in number of SOFs, to close a receive buffer when the last USB-OUT packet is an exact multiple of the endpoint's maximum packet size (see "USB-OUT packet flow" on page 715).

Table 500: FIFO Status and Control registers

Timing

CHAPTER 18

This chapter provides the electrical specifications, or timing, integral to the operation of the NS9360. Timing includes information about DC and AC characteristics, output rise and fall timing, and crystal oscillator specifications.

Electrical characteristics

The NS9360 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 ever exceeded.

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 501: 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 (see "DC electrical characteristics," beginning on page 757) are satisfied over these ranges.

Parameter	Symbol†	Rating	Unit
DC supply voltage	V_{DDA}	3.0 to 3.6	V
	V_{DDC} (core) V_{DDC} (PLL)	1.4 to 1.6 1.425 to 1.575	V
Maximum junction temperature	T_J	125	^oC

† V_{DDA}: Ratings of I/O cells for 3.3V interface
V_{DDC}: Ratings of internal cells

Table 502: Recommended operating conditions

Power dissipation

Table 503 shows the *maximum power dissipation* for I/O and core:

CPU clock		Full	No LCD	No LCD, no serial
Total @ 177 MHz		639 mW	599 mW	599 mW
	Core	276 mW	269 mW	269 mW
	I/O	363 mW	330 mW	330 mW
Total @ 155 MHz		593 mW	564 mW	564 mW
	Core	243 mW	237 mW	237 mW
	I/O	350 mW	327 mW	327 mW
Total @ 103 MHz		475 mW	450 mW	450 mW
	Core	164 mW	159 mW	159 mW
	I/O	311 mW	291 mW	291 mW

Table 503: NS9360 maximum power dissipation

Table 504 shows the *refresh only* power dissipation for I/O and core.

CPU clock		Refresh only
Total @ 177 MHz		255.5 mW
	Core	61.5 mW
	I/O	194 mW
Total @ 155 MHz		237.5 mW
	Core	54 mW
	I/O	183.5 mW
Total @ 103 MHz		197.4 mW
	Core	39 mW
	I/O	158.4 mW

Table 504: Refresh only mode

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: LVTTL level		Min	 2.0	 V
V_{IL}	Low-level input voltage: LVTTL level		Max	 0.8	 V
I_{IH}	High level input current (no pulldown) Input buffer with pulldown	V_{INA}=V_{DDA}	Min/Max Min/Max	-10/10 10/200	µA µA
I_{IL}	Low-level input current (no pullup Input buffer with pullup	V_{INA}=V_{SS}	Min/Max Min/Max	-10/10 10/200	µA µA
I_{OZ}	High-impedance leakage current	V_{OUTA}=V_{DDA} or V_{SS}	Min/Max	-10/10	µA

Table 505: DC electrical inputs

USB internal PHY DC electrical inputs

Symbol	Parameter	Min	Max	Units	Notes
V_{IH}	Input high level (driven)	2.0		V	
V_{IZ}	Input high level (floating)	2.7	3.6	V	
V_{IL}	Input low level		0.8	V	
V_{DI}	Differential input sensitivity	0.2		V	1

Table 506: USB internal PHY DC electrical inputs

Symbol	Parameter	Min	Max	Units	Notes
V_{CM}	Differential common mode range	0.8	2.5	V	2

Table 506: USB internal PHY 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

Table 507: DC electrical outputs

USB internal PHY 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 508: USB internal PHY DC electrical outputs

Notes:

1 Measured with R_L of 1.425k ohm to 3.6V.

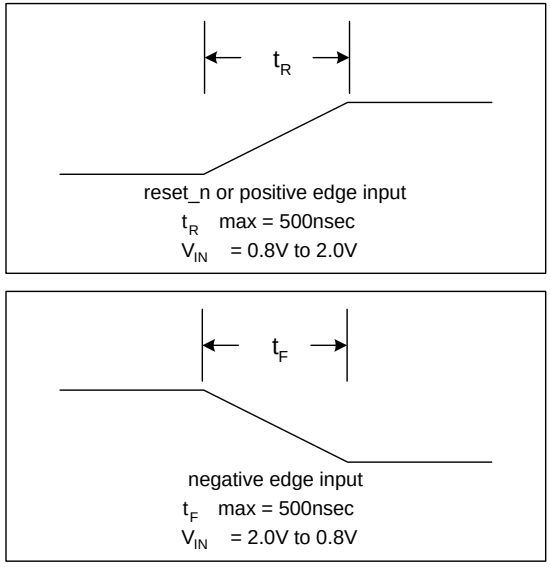
2 Measured with R_L of 14.25k ohm to GND.

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:

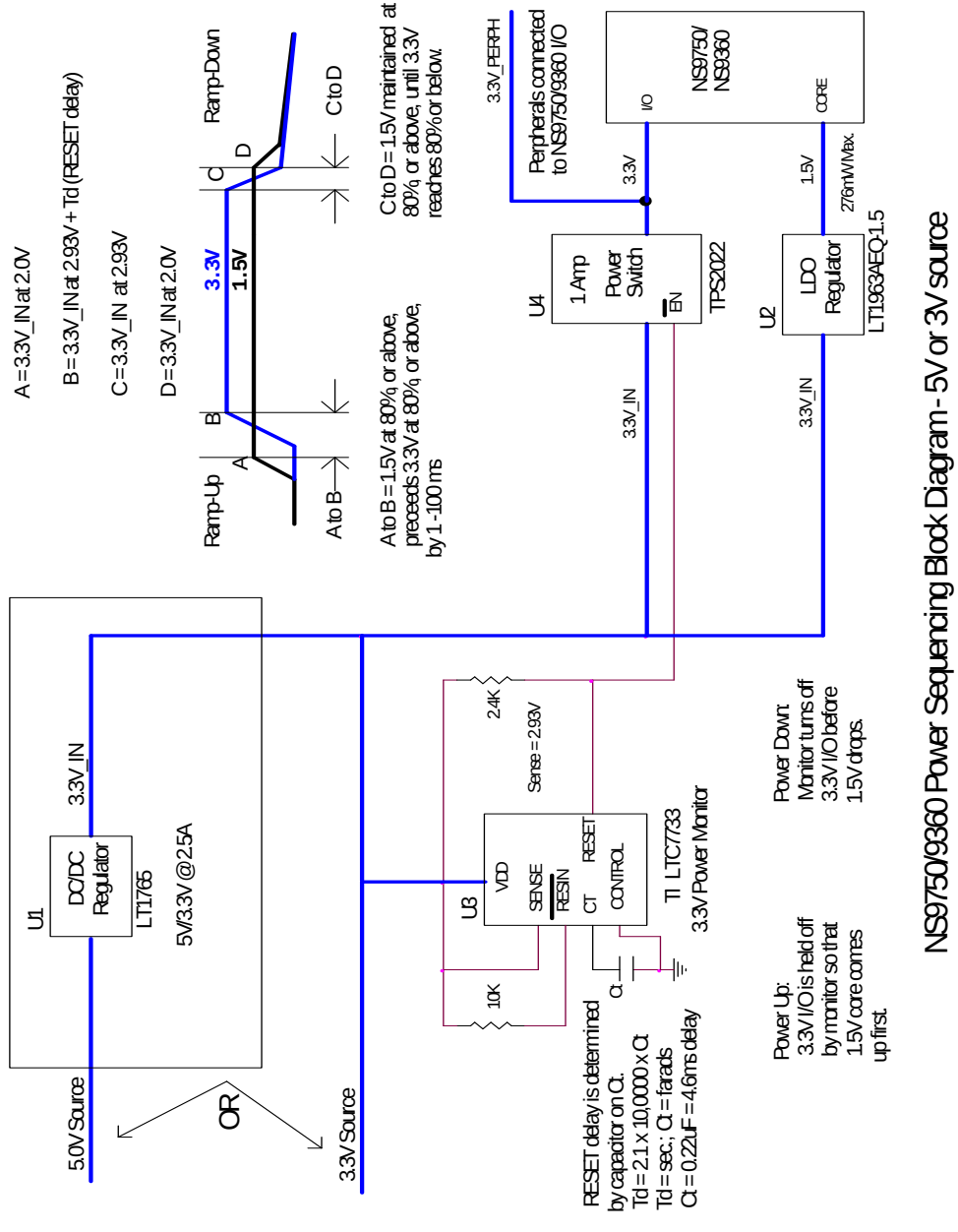


If an external device driving the reset or edge sensitive input on a Digi processor cannot meet the 500ns maximum rise and fall time requirement, the signal must be buffered with a Schmitt trigger device. Here are sample Schmitt trigger device part numbers:

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:



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. The next table describes the values shown in the SDRAM timing diagrams.

Parm	Description	Min	Max	Unit	Notes
M1	data input setup time to rising	1.0		ns	
M2	data input hold time to rising	0.0		ns	
M4	clk_out high to address valid		6.4	ns	
M11	address hold time	4.0			
M5	clk_out high to data_mask		6.4	ns	1, 2
M6	clk_out high to dy_cs_n low		6.4	ns	3, 4
M7	clk_out high to ras_n low		6.4	ns	
M8	clk_out high to cas_n low		6.4	ns	
M9	clk_out high to we_n low		6.4	ns	
M10	clk_out high to data out		6.6	ns	
M12	data out hold time	4.0			
M3	clk_out high to clk_en high		6.4	ns	
M13	clk_en high to sdram access	2	2	clock	
M14	end sdram access to clk_en low	2	2	clocks	

Table 509: 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)

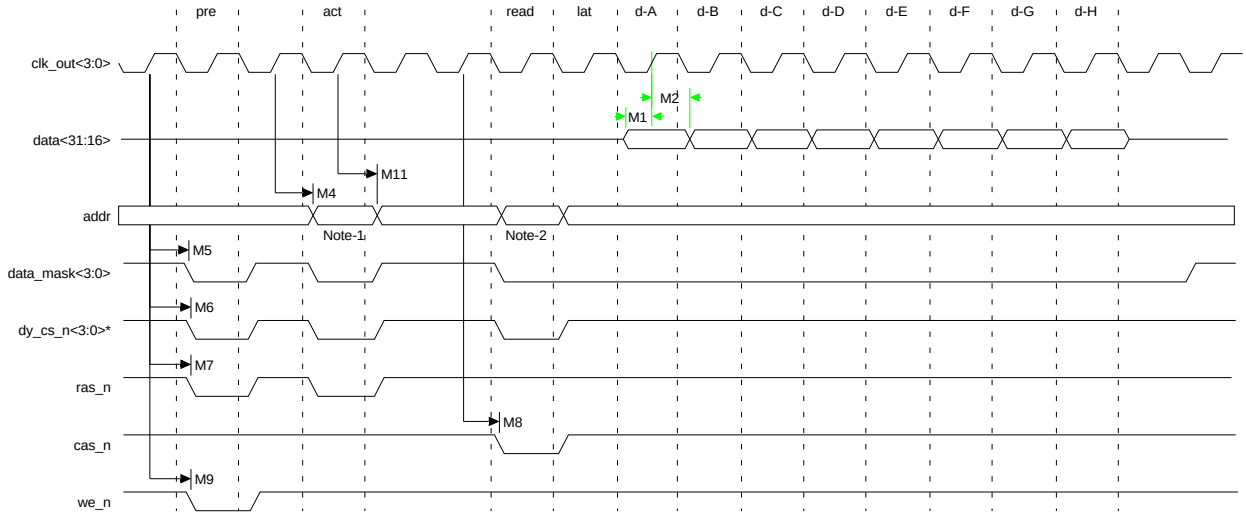


Figure 110: SDRAM burst read (16-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (16-bit), CAS latency = 3

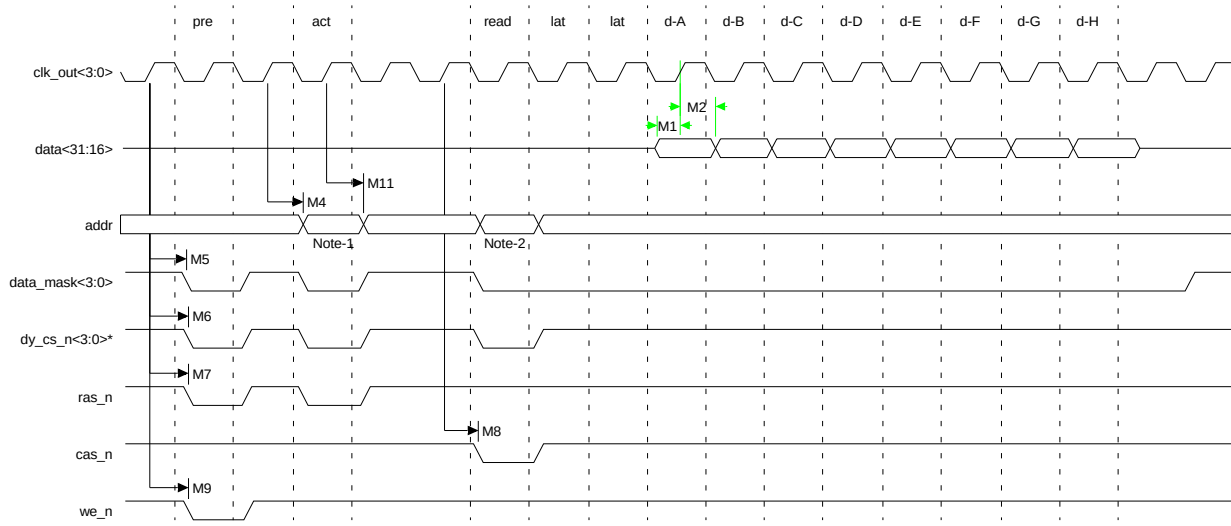


Figure 111: SDRAM burst read (16-bit), CAS latency = 3 timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst write (16-bit)

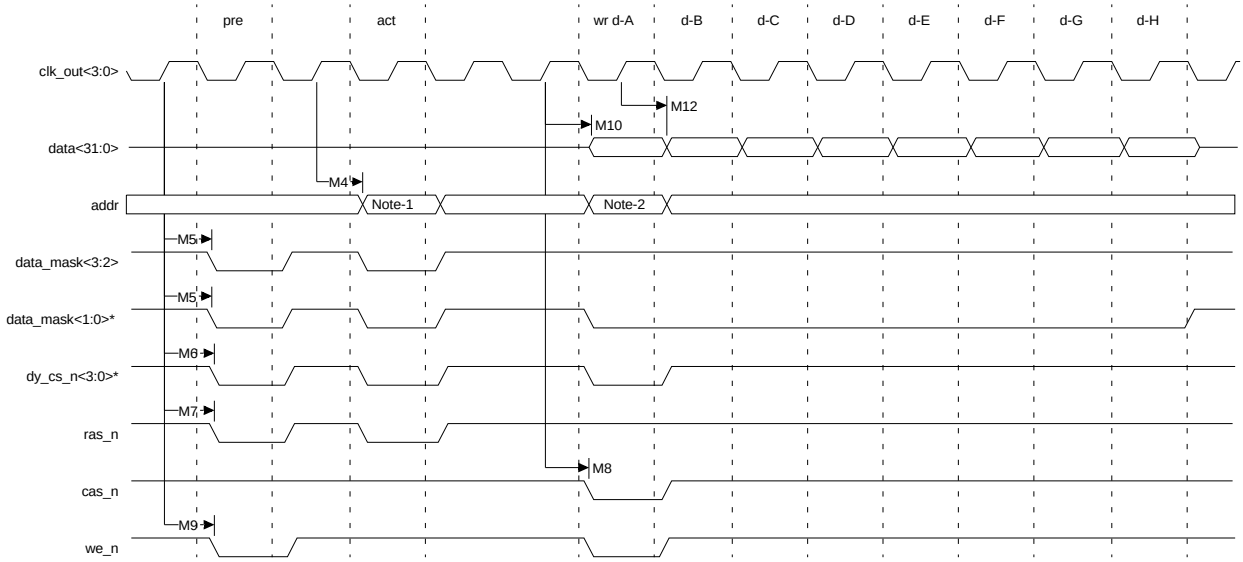


Figure 112: SDRAM burst write (16-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (32-bit)

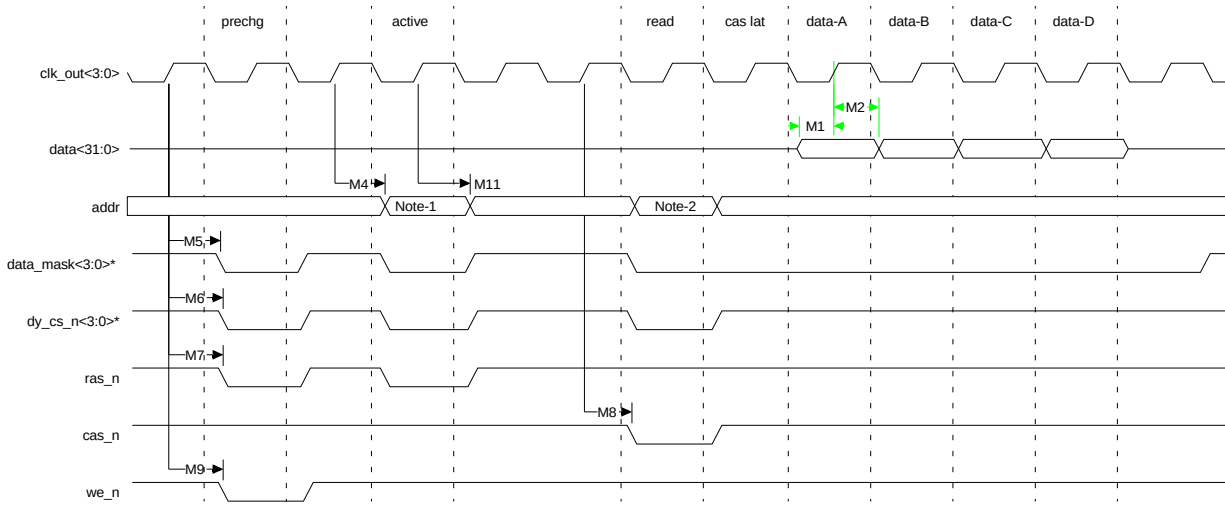


Figure 113: SDRAM burst read (32-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst read (32-bit), CAS latency = 3

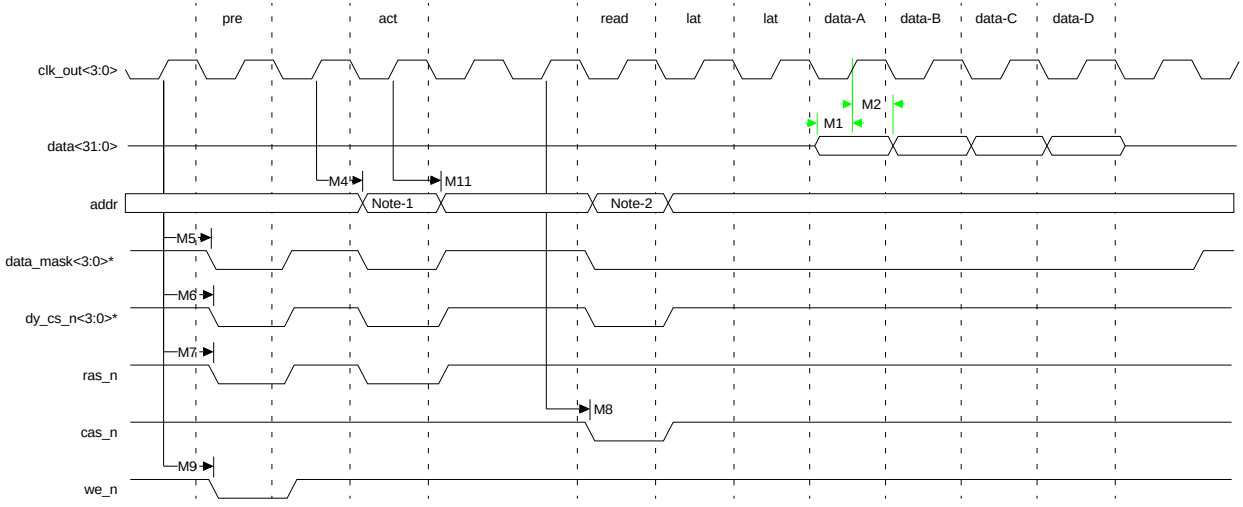


Figure 114: SDRAM burst read (32-bit), CAS latency = 3 timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM burst write (32-bit)

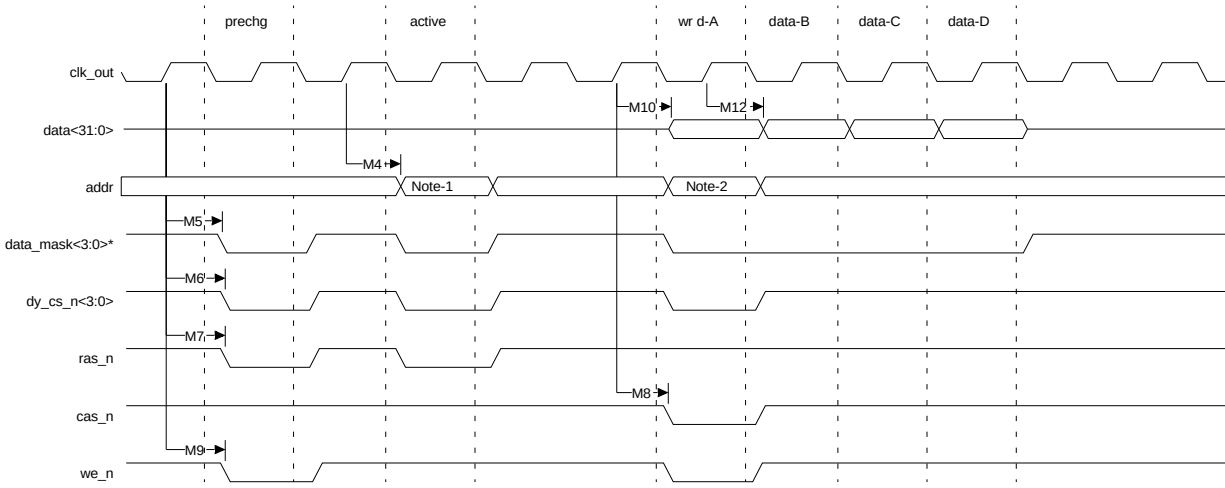


Figure 115: SDRAM burst write (32-bit) timing

Notes:

1 This is the bank and RAS address.

2 This is the CAS address.

SDRAM load mode

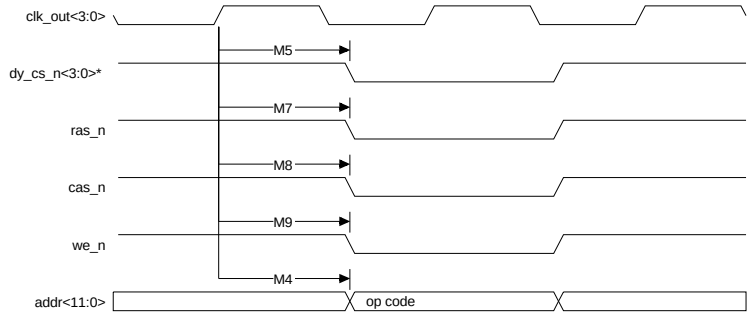


Figure 116: SDRAM load mode timing

SDRAM refresh mode

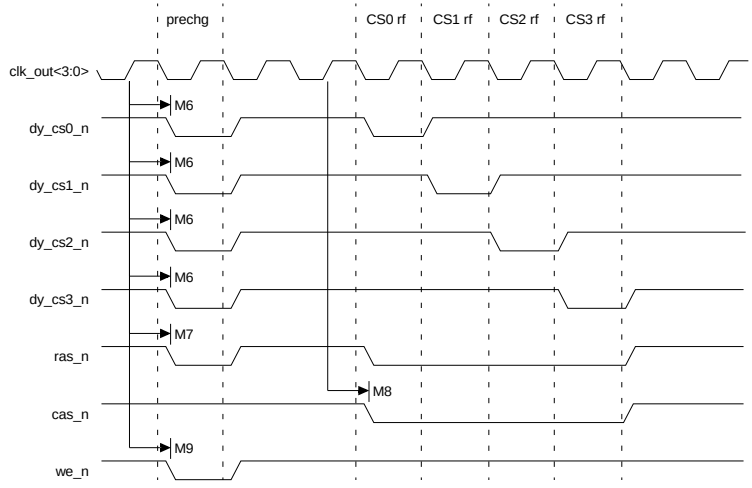


Figure 117: SDRAM refresh mode timing

Clock enable timing

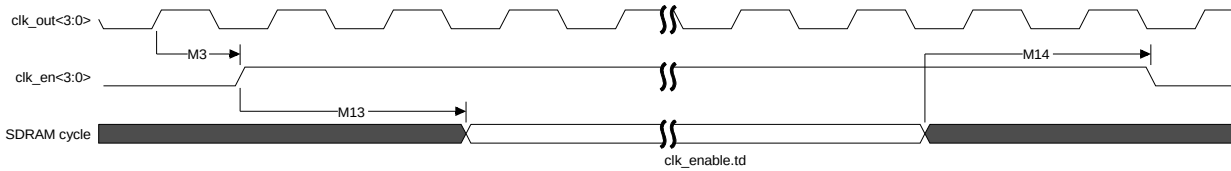


Figure 118: Clock enable timing

The next table describes the values shown in the SRAM timing diagrams.

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 510: 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 1 wait states

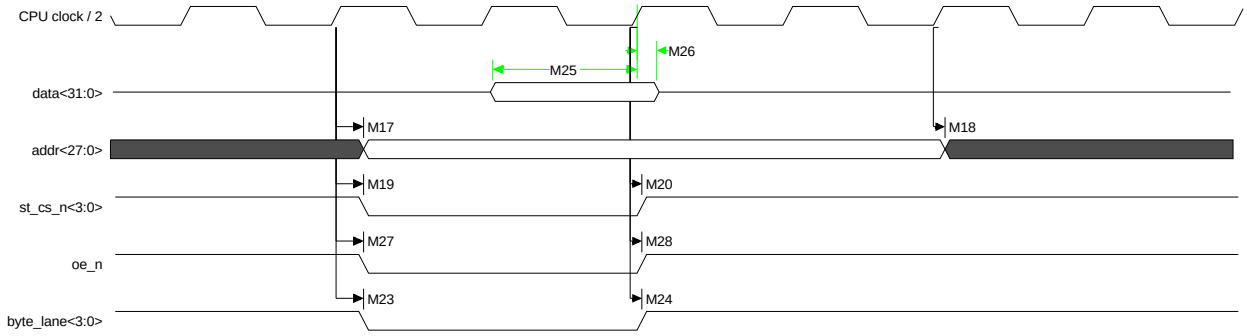


Figure 119: Static RAM read cycles with 1 wait state timing

- WTRD = 1

  WOEN = 0
- 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

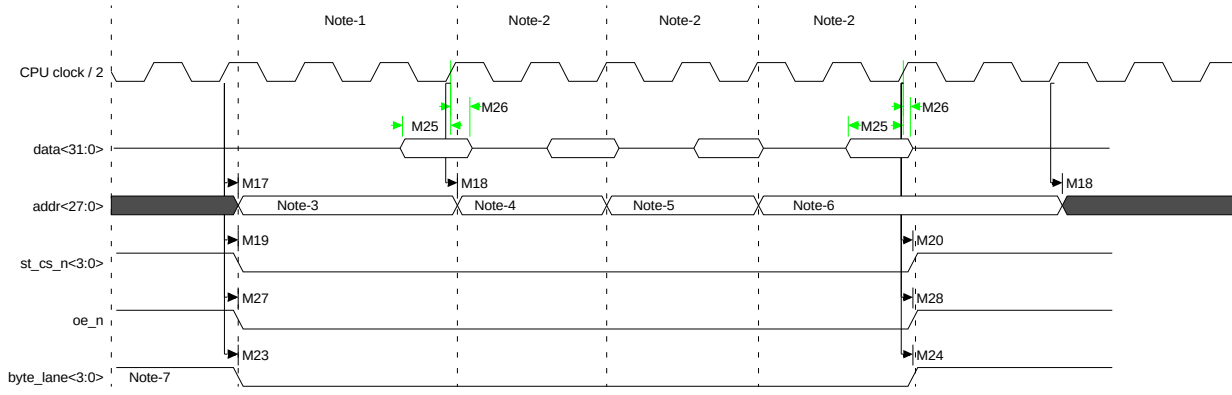


Figure 120: Static RAM asynchronous page mode read, WTPG = 1 timing

- 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

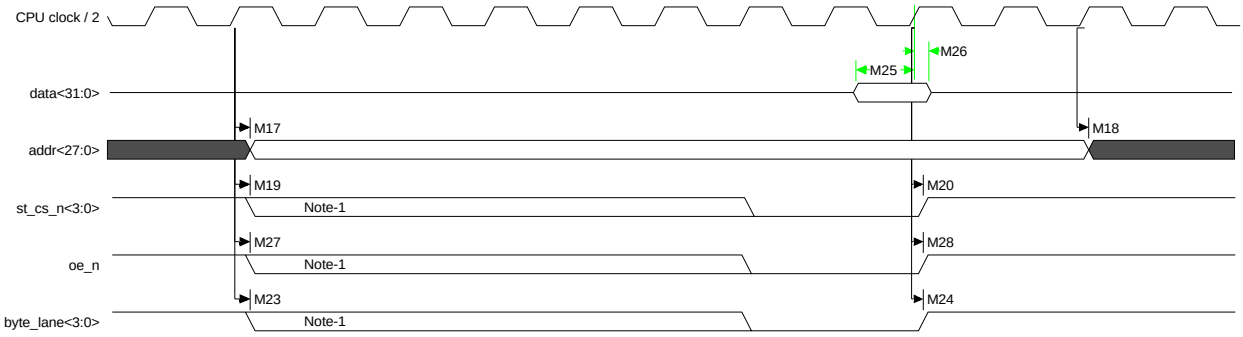


Figure 121: Static RAM read cycle with configurable wait states

- 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

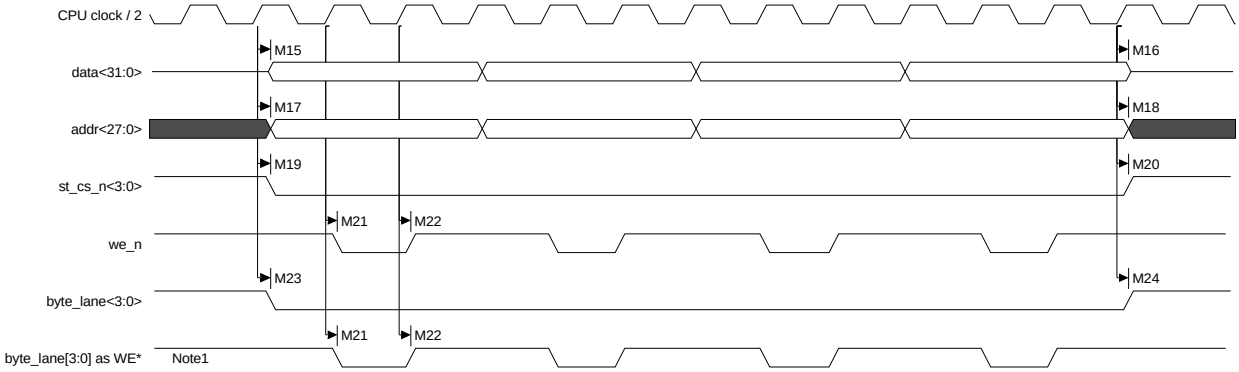


Figure 122: Static RAM sequential write cycles

- 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

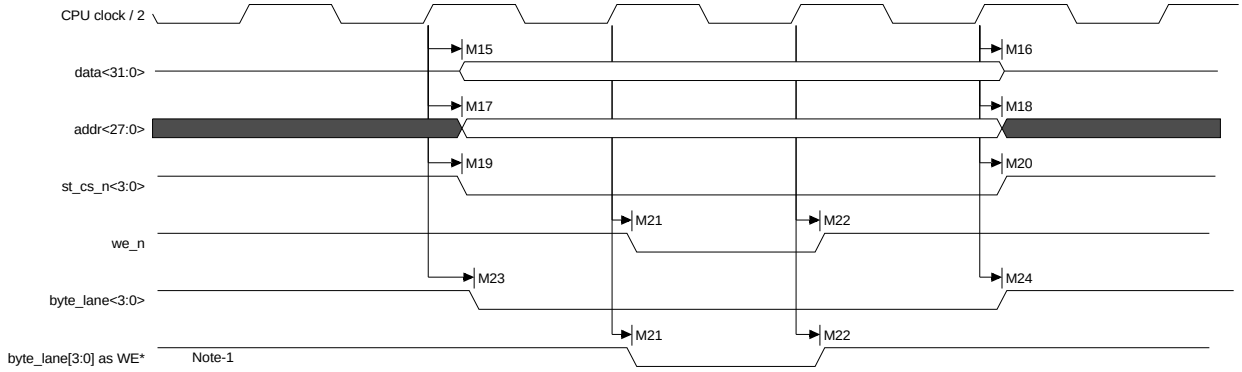


Figure 123: Static RAM write cycle

- 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

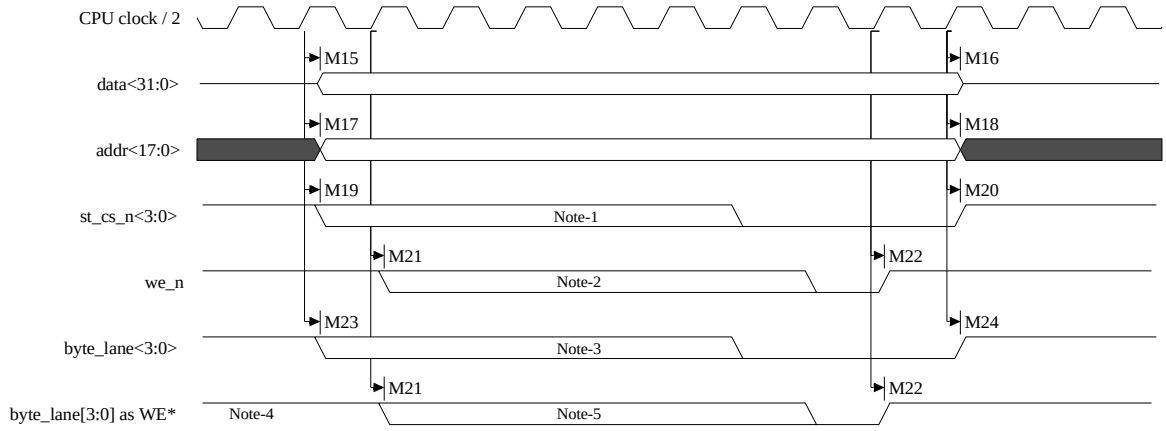


Figure 124: Static write cycle with configurable wait states

- 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.

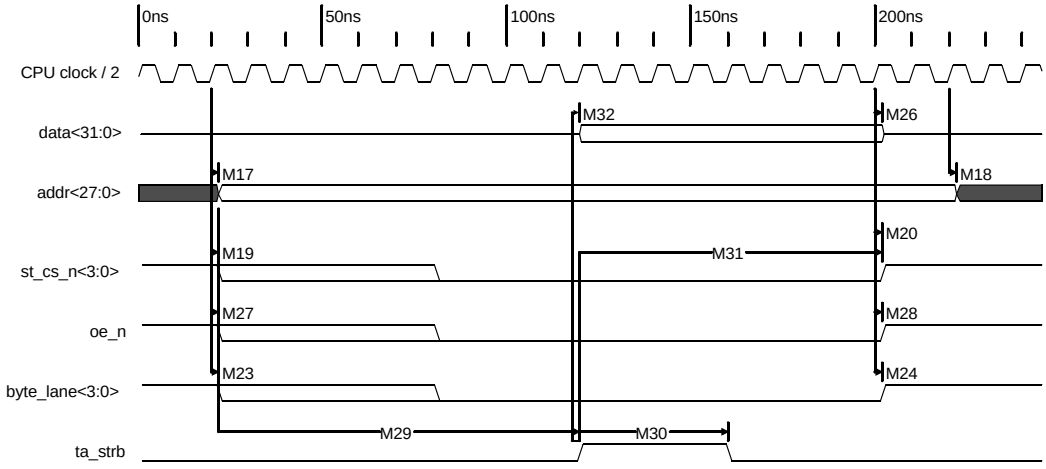
Slow peripheral acknowledge timing

Table 511 describes the values shown in the slow peripheral acknowledge timing diagrams.

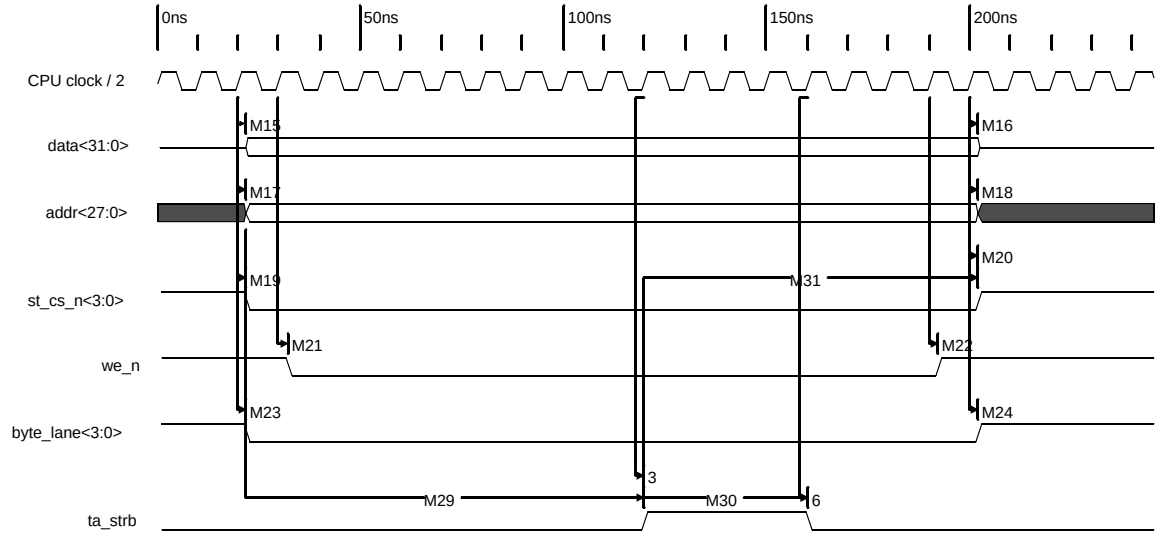
Parameter	Description	Min	Max	Unit	Notes
M15	clock high to data out valid	-2	+2	ns	
M16	data out hold time from clock high	-2	+2	ns	
M17	clock high to address valid	-2	+2	ns	
M18	address hold time from clock high	-2	+2	ns	
M19	clock high to st_cs_n low	-2	+2	ns	2
M20	clock high to st_cs_n high	-2	+2	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	
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	
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 511: Slow peripheral acknowledge timing parameters

Slow peripheral acknowledge read



Slow peripheral acknowledge write



Ethernet timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the Ethernet timing diagrams.

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)	20	36	ns	1,2
E7	mdc period	91		ns	4
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 512: Ethernet timing characteristics

Notes:

1 Minimum specification is for fastest AHB bus clock of 88.5 MHz. Maximum specification is for slowest AHB bus clock of 51.6 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.

4 Minimum specification is for fastest AHB clock at 88.5 MHz

Ethernet MII timing

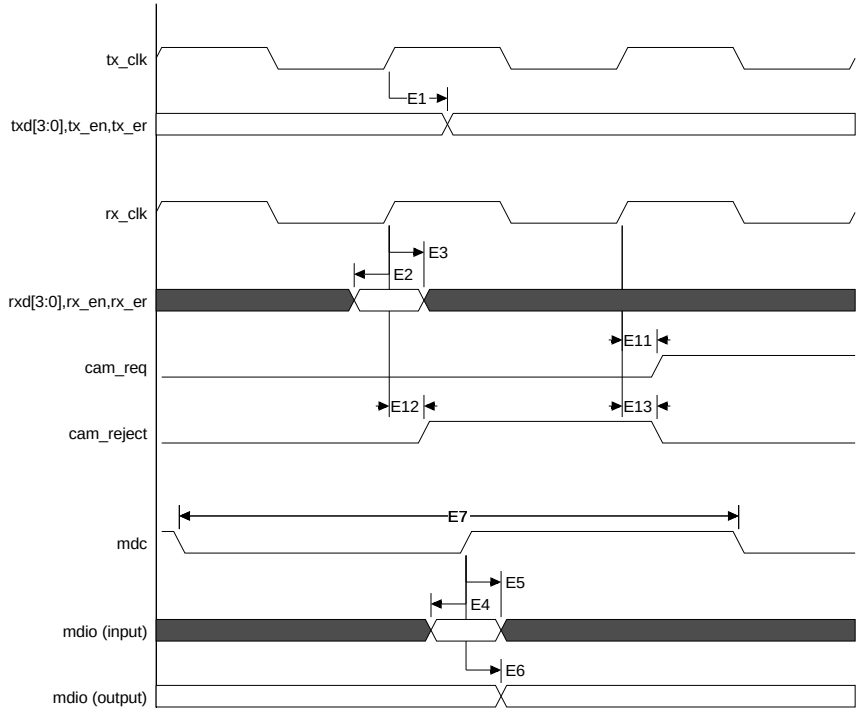


Figure 125: Ethernet MII timing

Ethernet RMII timing

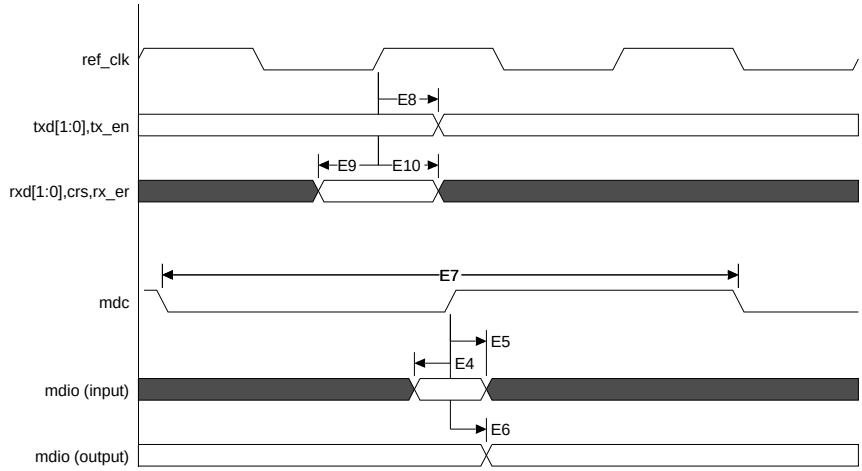


Figure 126: Ethernet RMII timing

I²C timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the I²C timing diagram.

		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 513: I²C timing parameters

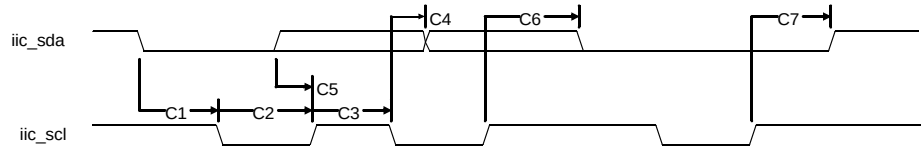


Figure 127: I²C timing

LCD timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the LCD timing diagrams.

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 notes 1 & 10)	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	LCDTiming1	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 514: LCD timing parameters

Parm	Description	Register	Value	Units
L18	CLCP to data/control (see notes 7 and 8)		-2.0 (min) +2.0 (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		22.22 (minimum)	ns

Table 514: 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 LCDTiming1 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)
- 18-bit TFT

6 See "LCDTiming0," beginning on page 543, and "LCDTiming1," beginning on page 546, 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 134, "LCD output timing," on page 788). 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 45 MHz.

Horizontal timing for STN displays

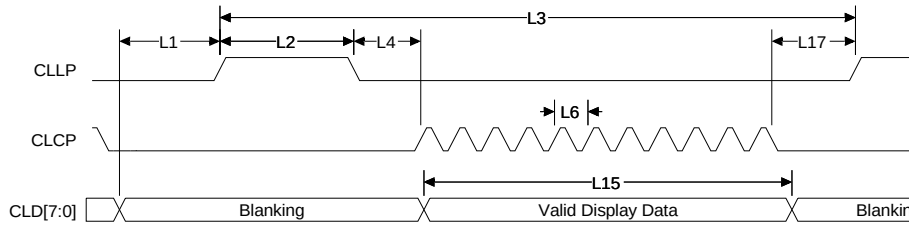


Figure 128: Horizontal timing for STN displays

Vertical timing for STN displays

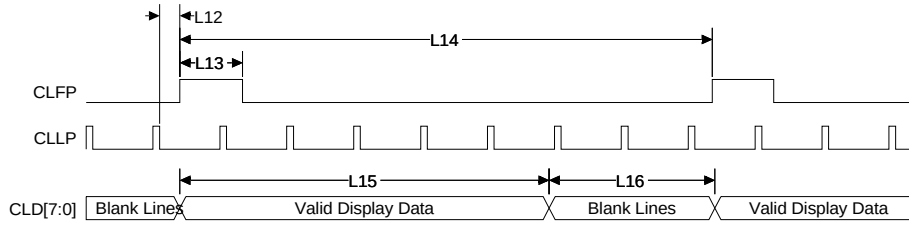


Figure 129: Vertical timing parameters for STN displays

Horizontal timing for TFT displays

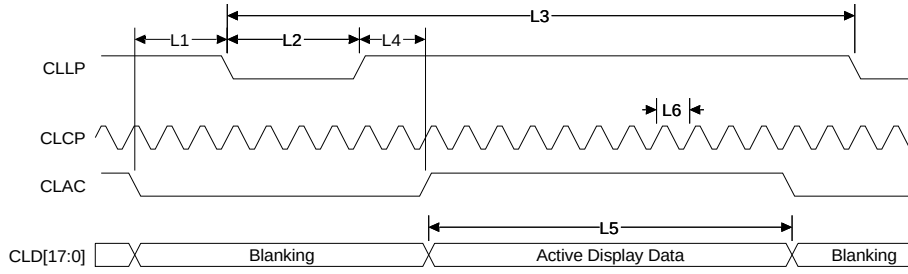


Figure 130: Horizontal timing parameters for TFT displays

Vertical timing for TFT displays

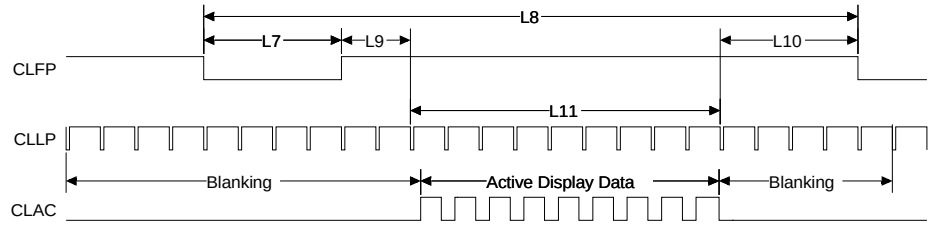


Figure 131: Vertical timing parameters for TFT displays

HSYNC vs VSYNC timing for STN displays

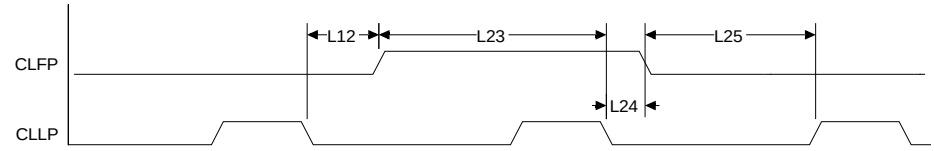


Figure 132: HSYNC vs VSYNC timing for STN displays

HSYNC vs VSYNC timing for TFT displays

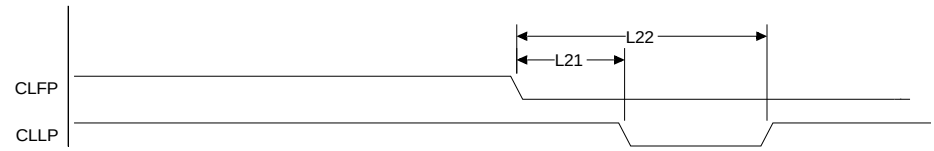


Figure 133: HSYNC vs VSYNC timing for TFT displays

LCD output timing

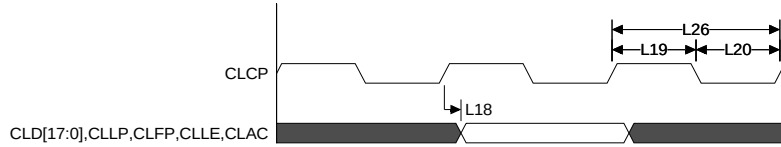


Figure 134: LCD output timing

SPI timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the SPI timing diagrams.

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}*4		ns	0, 1, 2, 3	3
SPI slave parameters						
SP14	SPI enable low setup to first SPI CLK in rising	30		ns	0, 3	1

Table 515: SPI timing parameters

Parm	Description	Min	Max	Units	Modes	Notes
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}*8		ns	0, 1, 2, 3	

Table 515: SPI timing parameters

Notes:

1 Active level of SPI enable is inverted (that is, 1) if the CSPOL bit in Serial Channel 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 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} = 5pf 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)

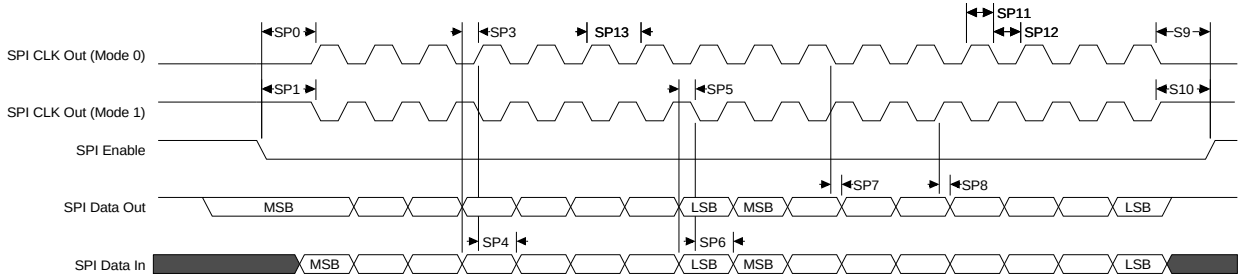


Figure 135: SPI master mode 0 and 1 (2-byte transfer)

SPI master mode 2 and 3: 2-byte transfer

(see note 7)

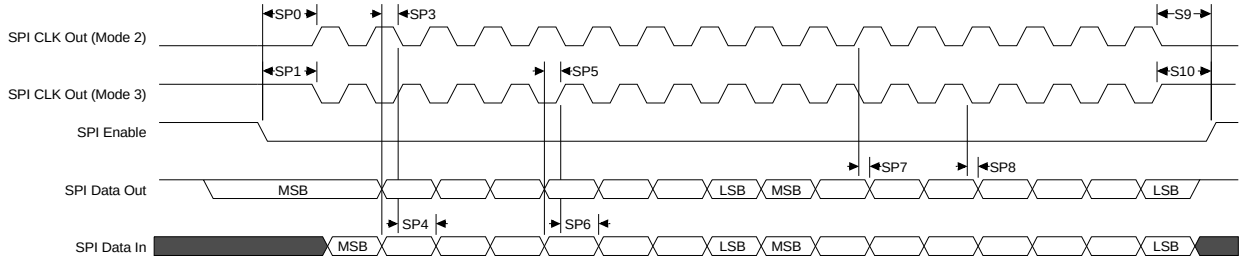


Figure 136: SPI master mode 2 and 3 (2-byte transfer)

SPI slave mode 0 and 1: 2-byte transfer

(see note 7)

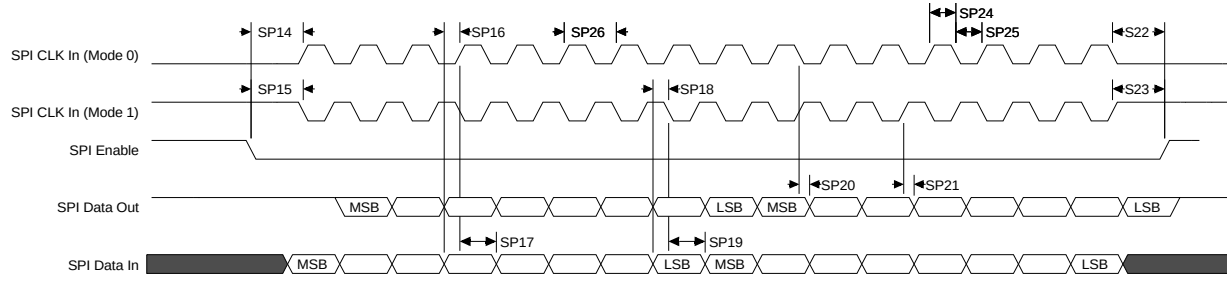


Figure 137: SPI slave mode 0 and 1 (2-byte transfer)

SPI slave mode 2 and 3: 2-byte transfer

(see note 7)

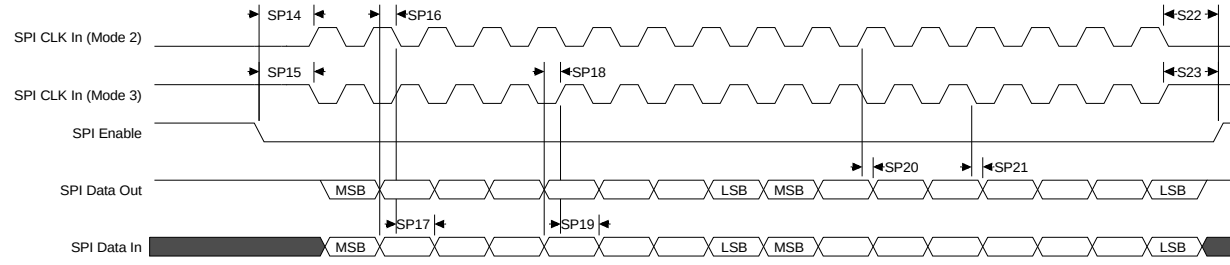


Figure 138: SPI slave mode 2 and 3 (2-byte transfer)

IEEE 1284 timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the IEEE 1284 timing diagram.

Parm	Description	Min	Max	Unit	Note
IE1	Busy-while-Strobe	0	511	ns	1
IE2	Busy high to nAck low	0		ns	
IE3	Busy high		1022	ns	2
IE4	nAck low		511	ns	3
IE5	nAck high to Busy low		511	ns	3

Table 516: IEEE 1284 timing parameters

Notes:

1 The range is 0ns up to one time unit.

2 Two time units.

3 One time unit.

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 45 MHz and the Granularity Count register is set to 23. The basic time unit is 1/45 MHz x 23, which is 511ns (the basic time unit must be at least 511ns).

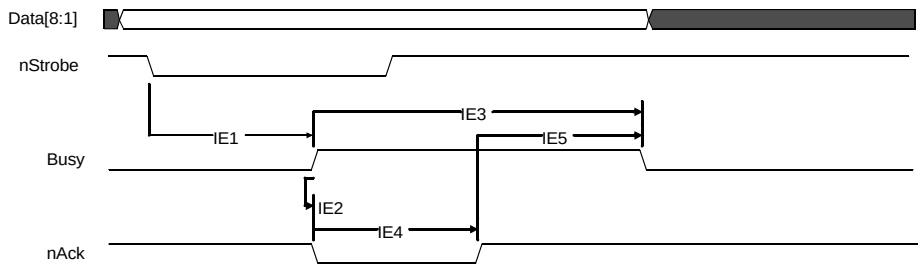


Figure 139: IEEE 1284 timing with BBus clock at 50 MHz and GCR set to 25

USB internal PHY timing

The next two tables describe the values shown in the USB internal PHY timing diagrams.

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	%	1, 2, 5
U4	Driver output resistance	28	44	ohms	3

Table 517: USB internal PHY 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, 4, 5

Table 518: USB internal PHY low speed timing parameters

Notes:

1 Load shown in Figure 141, "USB internal PHY full speed load," on page 795.

2 U1/U2.

3 Includes resistance of 27 ohm ± 2 ohm external series resistor.

4 Load shown in Figure 142, "USB internal PHY low speed load," on page 796.

5 Excluding the first transition from the idle state.

USB internal PHY differential data timing

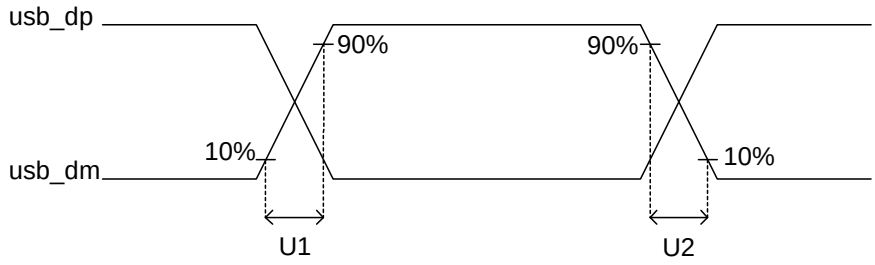


Figure 140: USB internal PHY differential data

USB internal PHY full speed load timing

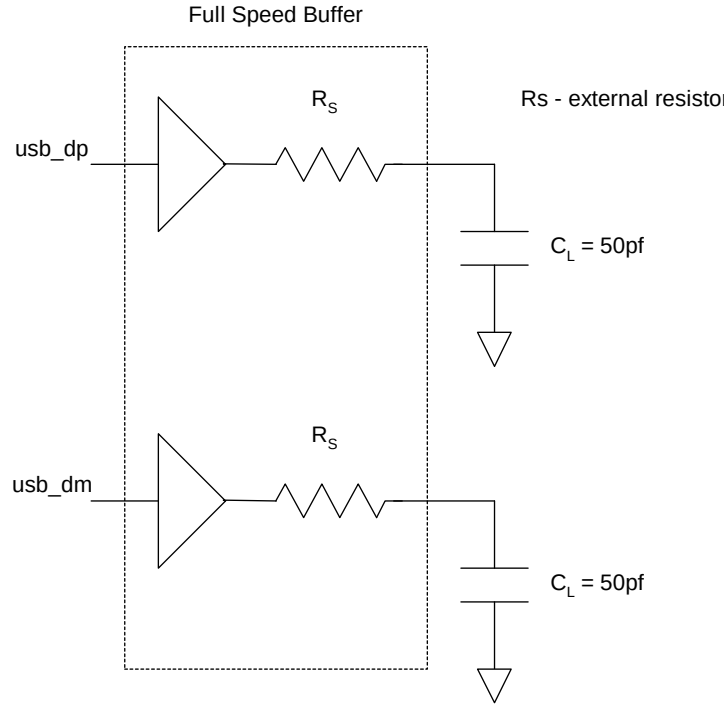


Figure 141: USB internal PHY full speed load

USB internal PHY low speed load

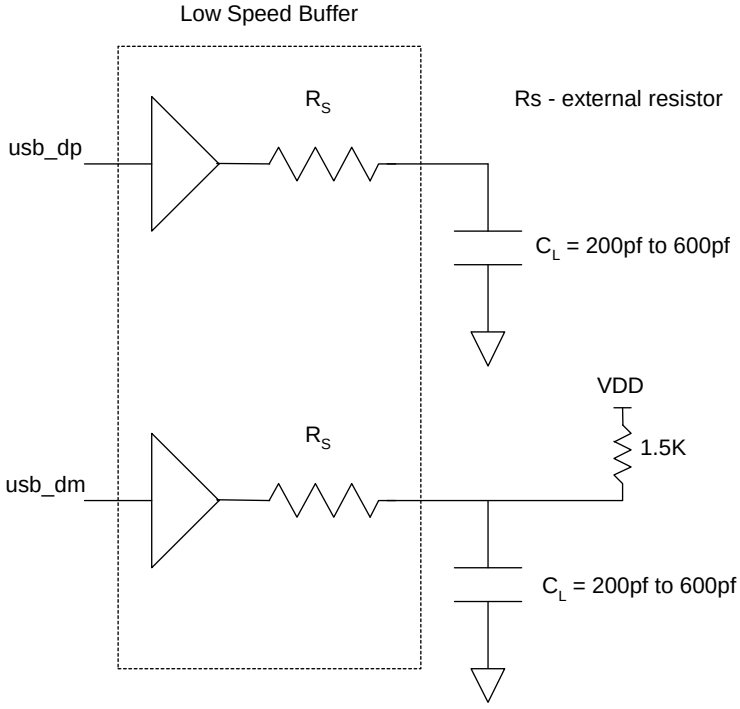


Figure 142: USB internal PHY low speed load

USB external PHY interface

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

Table 519 describes the values shown in the USB external PHY interface output timing diagram (Figure 143).

Parm	Description	Min	Max	Unit	Notes
U5	Output skew	-0.25	0.25	ns	1

Table 519: USB external PHY interface output timing parameters

Note:

1 Output skew between any of the output signals that interface to an external USB PHY; that is, GPIO[44:42] and GPIO[49:48] for primary interface or GPIO [52:50] and GPIO[55:54] for duplicate interface.

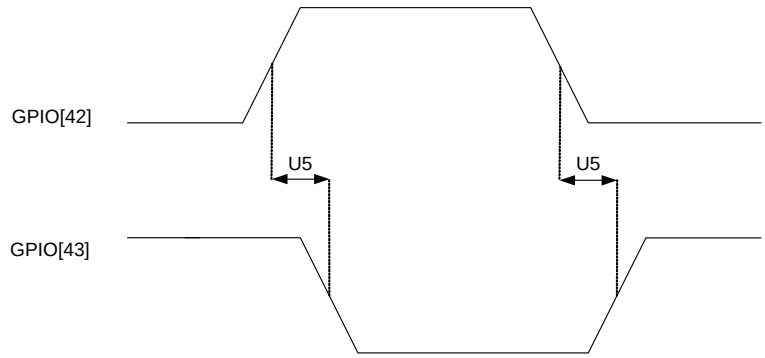


Figure 143: USB external PHY interface output timing

Reset and hardware strapping timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the reset and hardware strapping timing diagram.

Parm	Description	Min	Typ	Unit	Notes
R1	reset_n minimum time	10		x1_sys_osc clock cycles	1
R2	reset_n to reset_done		NOR flash: 4.5 SPI flash: 15	ms	

Table 520: Reset and hardware strapping timing parameters

Note:

1 The hardware strapping pins are latched 5 clock cycles after reset_n is deasserted (goes high).

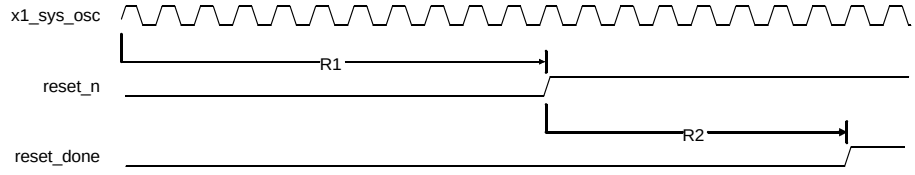


Figure 144: Reset and hardware strapping timing

JTAG timing

Note: All AC characteristics are measured with 10pF, unless otherwise noted.

The next table describes the values shown in the JTAG timing diagram.

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 521: JTAG timing parameters

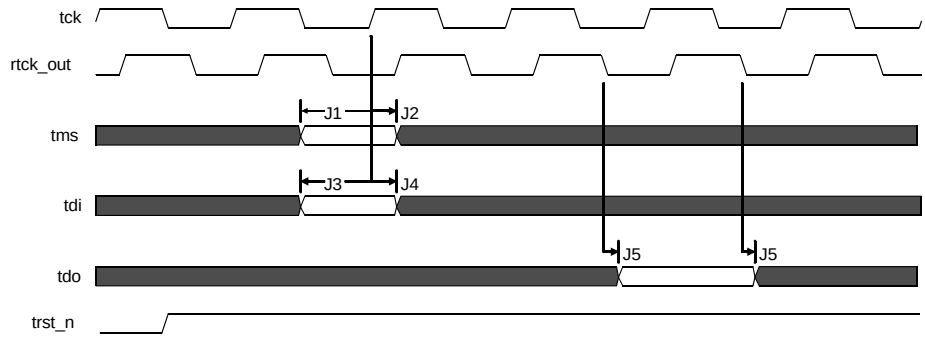


Figure 145: JTAG timing

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

The next table describes the values shown in the USB crystal/external oscillator timing diagram.

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 522: USB crystal/external oscillator timing parameters

Note:

1 If using a crystal, the tolerance must be ±100 ppm or better.

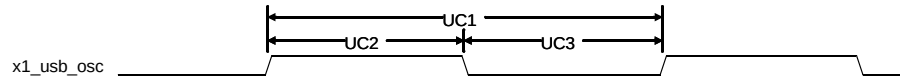


Figure 146: USB crystal/external oscillator timing

LCD input clock timing

The next table describes the values shown in the LCD input clock timing diagram.

Parm	Description	Min	Max	Unit	Notes
LC1	lcdclk cycle time	11.11		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 523: LCD input clock timing parameters

Note:

1 The clock rate supplied on lcdclk is twice the actual LCD clock rate.

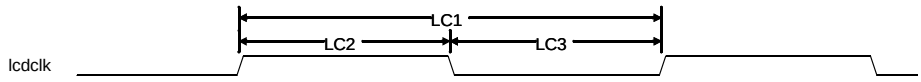


Figure 147: LCD input clock timing

System PLL reference clock timing

The next table describes the values shown in the system PLL reference clock timing diagram.

Parm	Description	Min	Max	Unit	Notes
SC1	x1_sys_osc cycle time	25	50	ns	
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 524: System PLL reference clock timing parameters

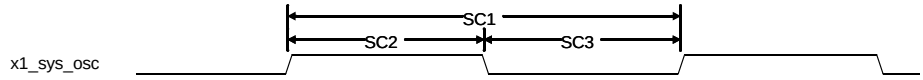


Figure 148: System PLL reference clock timing

Packaging

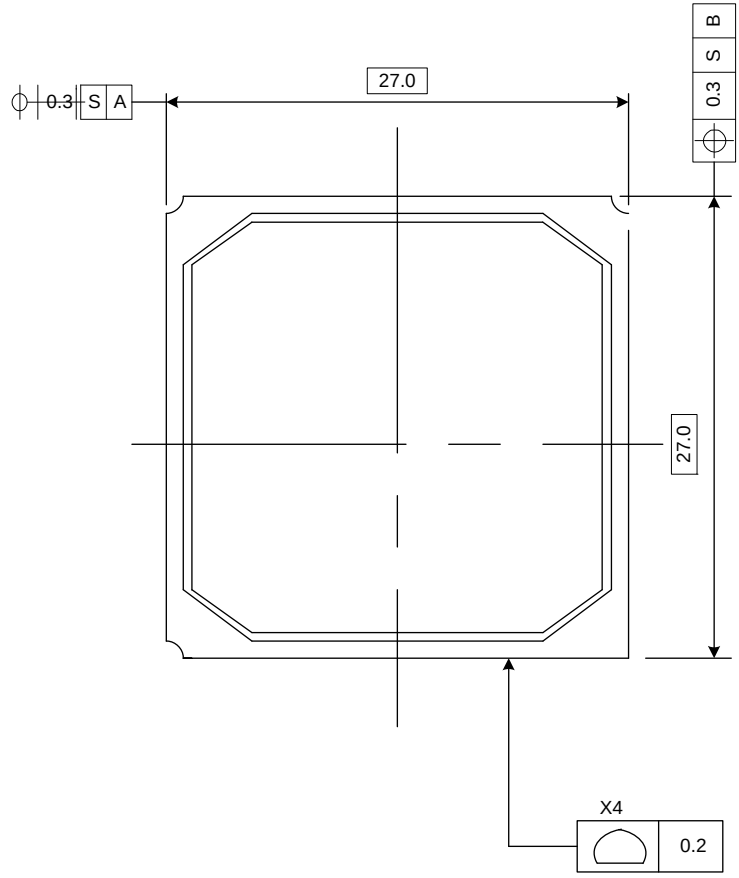
CHAPTER 19

The NS9360 is a complete system-on-chip processor, and includes Ethernet, display support, and a robust peripheral set.

NS9360 dimensions and pinout are shown on the following pages. The first set of diagrams shows the top, bottom, and side views, with dimensions. The next diagram is the NS9360 BGA layout.

The recommended pad size for this package configuration is 0.60 mm.

Top view—NS9360



Side and bottom view—NS9360

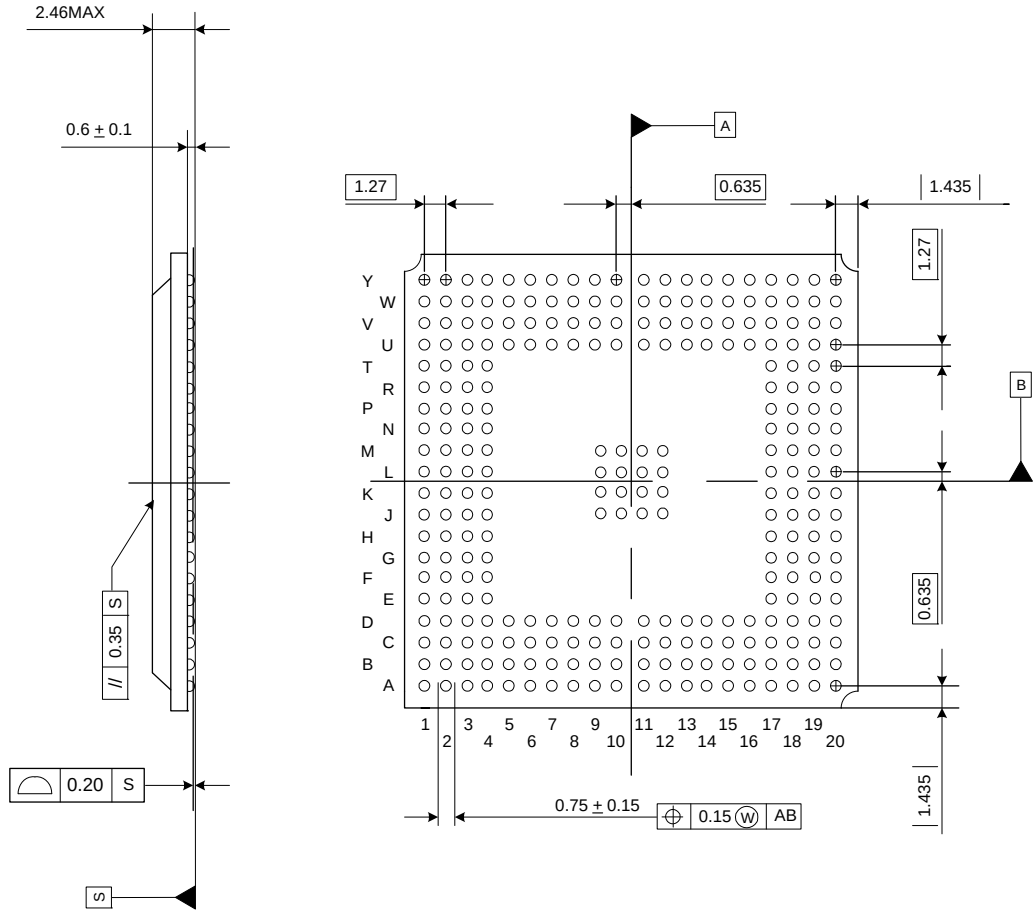


Figure 149 shows the layout of the NS9360, for use in setting up the board.

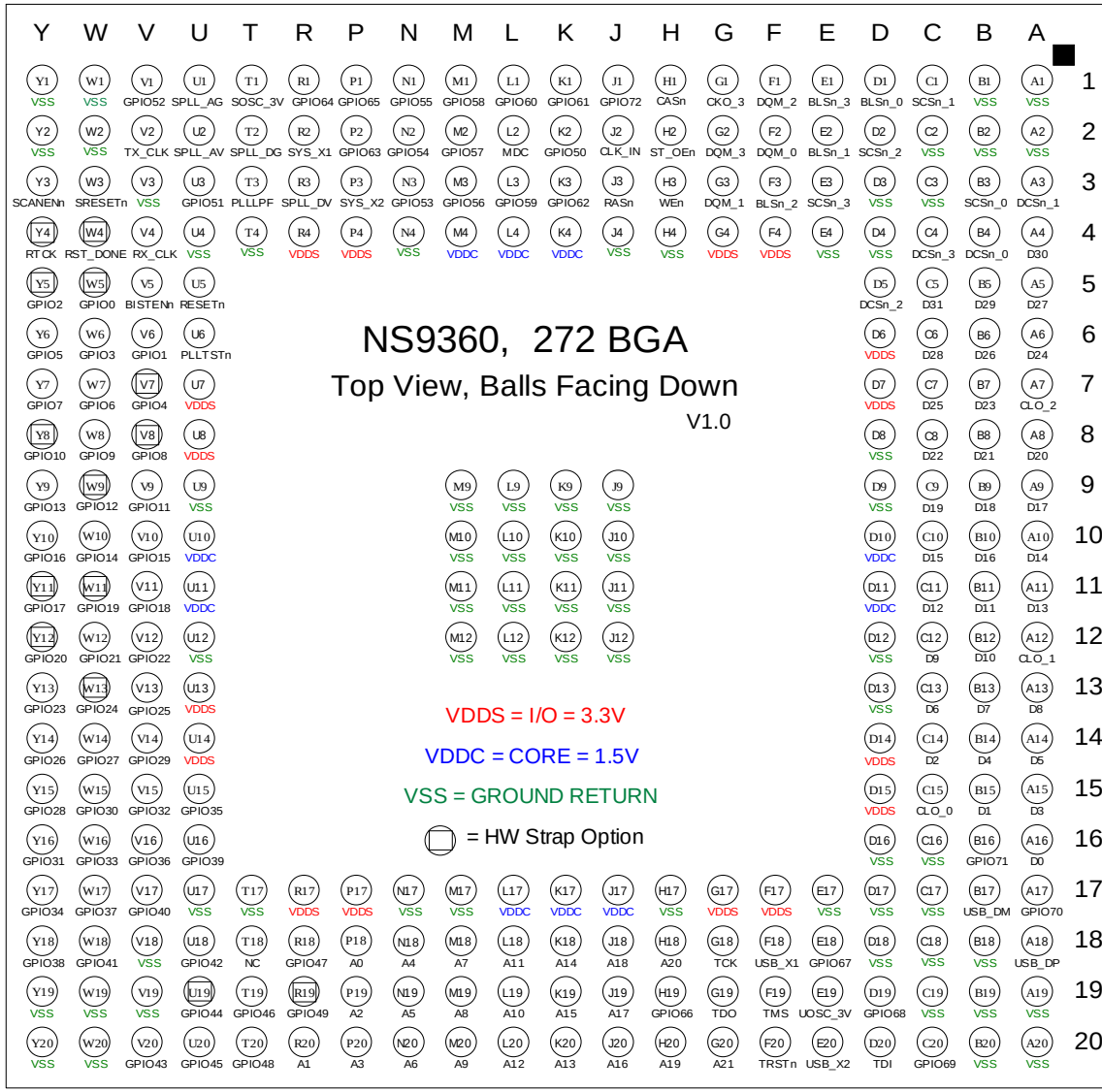


Figure 149: NS9360 BGA layout

For information about hardware strapping options, see the bootstrap initialization configuration pins in Chapter 4, “System Control Module.”

Product specifications

These tables provide additional information about the NS9360.

RoHS substance	PPM level
Lead	0
Mercury	0
Cadmium	0
Hexavalent Chromium	0
Polybrominated biphenyls	0
Polybrominated diphenyl ethers	0

Table 525: NS9360 RoHS specifications

Component	Weight [mg]	Material		Weight [mg]	Content [%]
		CAS no.	Name		
Chip	7.20	7440-21-3	Silicon	6.080	0.239
		n/a	PI	1.120	0.044
Frame	650.70	n/a	BT resin	305.828	12.011
		7440-50-8	Copper	260.280	10.222
		7440-02-0	Nickel	3.254	0.128
		7440-57-5	Gold	0.651	0.026
		n/a	Brominated epoxy	65.070	2.555
		n/a	Other	15.617	0.613
Die attach	5.20	7440-22-4	Silver	4.160	0.163
		n/a	Epoxy resin	1.040	0.041
Wire bond	3.20	7440-57-5	Gold	3.200	0.126

Table 526: NS9360 materials sheet

Component	Weight [mg]	Material		Weight [mg]	Content [%]
Mold resin	1460.00	7440-21-3	Silica	1168.000	45.870
		n/a	Epoxy resin	116.800	4.587
		n/a	Phenol resin	87.600	3.440
		n/a	Other	87.600	3.440
Solder ball	420.00	7440-31-5	Tin	420.000	16.495
Total weight	2546.30			2546.300	100.000

Table 526: NS9360 materials sheet

Index

Some registers and terms are followed by a code in parentheses to indicate which module they are in. Some are self explanatory (for example, BBus DMA); for the others, use the following legend:

scm = System Control Module
mem = Memory Controller
MMU = memory management unit
USBH = USB Host module
USBD = USB Device module
1284 = IEEE 1284 module
ARM9 = ARM9 processor

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W


Digi®