MANCHESTER 1824

The University of Manchester

SpiNNaker datasheet version 2.01
19 October 2010

SpiNNaker - a chip multiprocessor for neural network simulation. Datasheet.

Features

- *18* ARM968 processors, each with:
  - *64* Kbytes of tightly-coupled data memory;
  - *32* Kbytes of tightly-coupled instruction memory;
  - DMA controller;
  - communications controller;
  - vectored interrupt controller;
  - low-power ‘wait for interrupt’ mode.
- Multicast communications router
  - *6* self-timed inter-chip bidirectional links;
  - *1,024* associative routing entries.
- Interface to 128Mbyte (nominal) Mobile DDR SDRAM
  - over 1 Gbyte/s sustained block transfer rate;
  - optionally incorporated within the same multi-chip package.
- Ethernet interface for host connection
- Fault-tolerant architecture
  - defect detection, isolation, and function migration.
- Boot, test and debug interfaces.

Introduction

SpiNNaker is a chip multiprocessor designed specifically for the real-time simulation of large-scale spiking neural networks. Each chip (along with its associated SDRAM chip) forms one node in a scalable parallel system, connected to the other nodes through self-timed links.

The processing power is provided through the multiple ARM cores on each chip. In the standard model, each ARM models multiple neurons, with each neuron being a coupled pair of differential equations modelled in continuous ‘real’ time. Neurons communicate through atomic ‘spike’ events, and these are communicated as discrete packets through the on- and inter-chip communications fabric. The packet contains a routing key that is defined at its source and is used to implement multicast routing through an associative router in each chip.

One processor on each SpiNNaker chip will perform system management functions; the communications fabric supports point-to-point packets to enable co-ordinated system management across local regions and across the entire system, and nearest-neighbour packets are used for system flood-fill boot operations and for chip debug. In addition, fixed-route packets carry 64 bits of debug information back to particular nodes for transmission to the host computer.

Background

SpiNNaker was designed at the University of Manchester within an EPSRC-funded project in collaboration with the University of Southampton, ARM Limited and Silistix Limited. Subsequent development took place within a second EPSRC-funded project which added the universities of Cambridge and Sheffield to the collaboration. The work would not have been possible without EPSRC funding, and the support of the EPSRC and the industrial partners is gratefully acknowledged.

Intellectual Property rights

Disclaimer

The details in this datasheet are presented in good faith but no liability can be accepted for errors or inaccuracies. The design of a complex chip multiprocessor is a research activity where there are many uncertainties to be faced, and there is no guarantee that a SpiNNaker system will perform in accordance with the specifications presented here.

The APT group in the School of Computer Science at the University of Manchester was responsible for all of the architectural and logic design of the SpiNNaker chip, with the exception of synthesizable components supplied by ARM Limited and interconnect components supplied by Silistix Limited. All design verification was also carried out by the APT group. As such the industrial project partners bear no responsibility for the correct functioning of the device.

Error notification and feedback

Please email details of any errors, omissions, or suggestions for improvement to:
steve.furber@manchester.ac.uk.

Change history

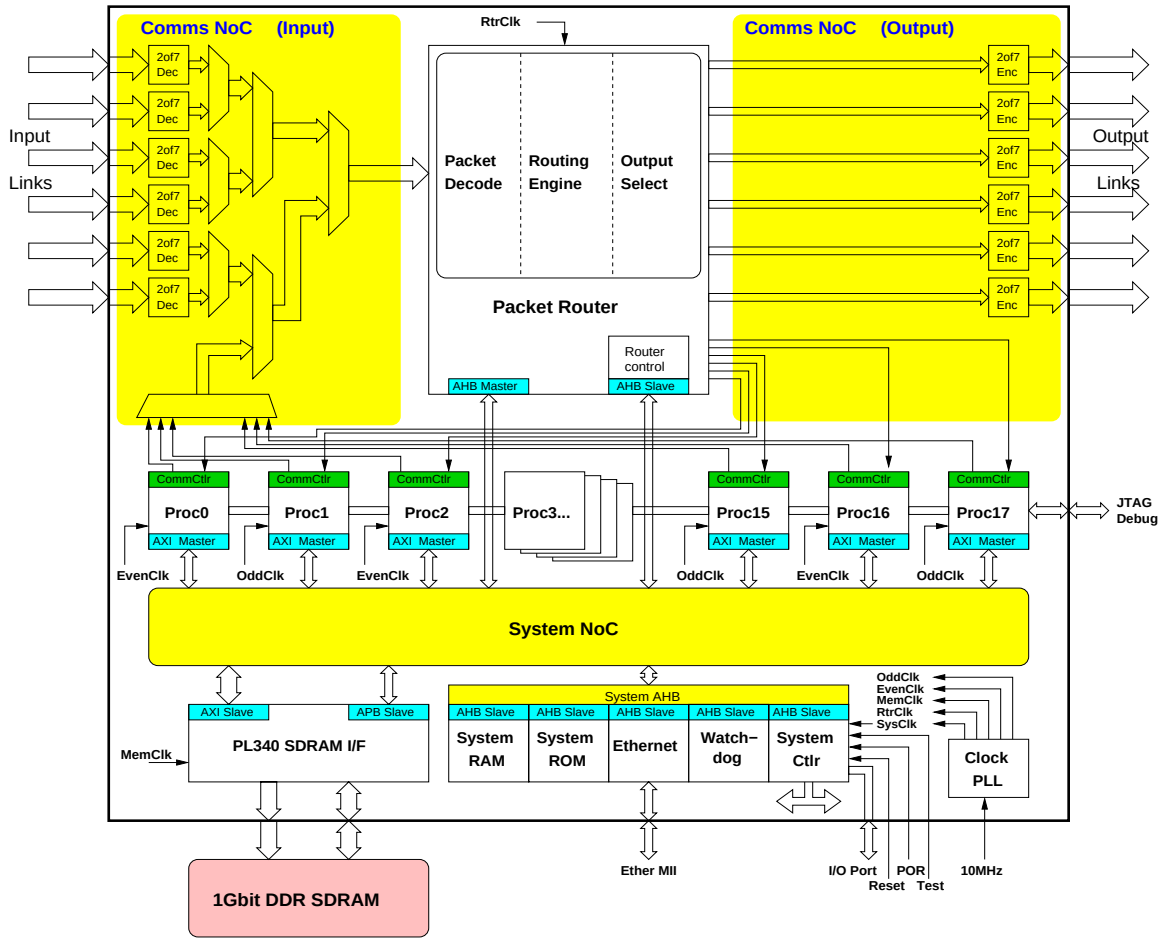
version	date	changes
2.00	21/4/10	Full SpiNNaker chip initial version
2.01	19/10/10	Change CPU clocks, add package details, minor corrections.

Contents

1. Chip organization

1.1 Block diagram

The primary functional components of SpiNNaker are illustrated in the figure below.



Each chip contains *18* identical processing subsystems. At start-up, following self-test, one of the processors is nominated as the Monitor Processor and thereafter performs system management tasks. The other processors are responsible for modelling one or more neuron fascicles - a fascicle being a group of neurons with associated inputs and outputs (although some processors may be reserved as spares for fault-tolerance purposes).

The Router is responsible for routing neural event packets both between the on-chip processors and from and to other SpiNNaker chips. The Tx and Rx interface components are used to extend the on-chip communications NoC to other SpiNNaker chips. Inputs from the various on- and off-chip sources are assembled into a single serial stream which is then passed to the Router.

Various resources are accessible from the processor systems via the System NoC. Each of the processors has access to the shared off-chip (but possibly in the same package) SDRAM, and various system components also connect through the System NoC in order that, whichever processor is Monitor Processor, it will have access to these components.

The sharing of the SDRAM is an implementation convenience rather than a functional requirement, although it may facilitate function migration in support of fault-tolerant operation.

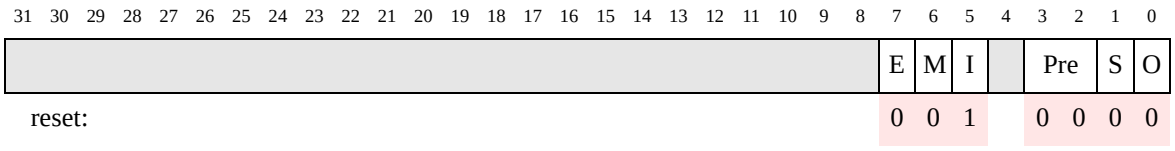
1.2 System-on-Chip hierarchy

The SpiNNaker chip is viewed as having the following structural hierarchy, which is reflected throughout the organisation of this datasheet:

- ARM968 processor subsystem
  - the ARM968, with its tightly-coupled instruction and data memories
  - Timer/counter and interrupt controller
  - DMA controller, including System NoC interface
  - Communications controller, including Communications NoC interface
- Communications NoC
  - Router, including multicast, point-to-point, nearest-neighbour, fixed-route, default and emergency routing functions
  - *6* bidirectional inter-chip links
  - communications NoC arbiter and fabric
- System NoC
  - SDRAM interface
  - System Controller
  - Router configuration registers
  - Ethernet MII interface
  - Boot ROM
  - System RAM
- Boot, test and debug
  - central controller for ARM968 JTAG functions
  - an off-chip serial boot ROM can be used if required

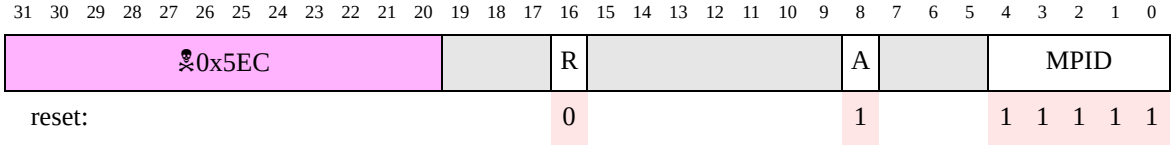
1.3 Register description convention

Registers are 32-bits (1 word) and are usually displayed in this datasheet as shown below:



- The grey-shaded areas of the register are unused. They will generally read as 0, and should be written as 0 for maximum compatibility with any future functionality extensions.
- Reset values, where defined, are shown against a red shaded background.

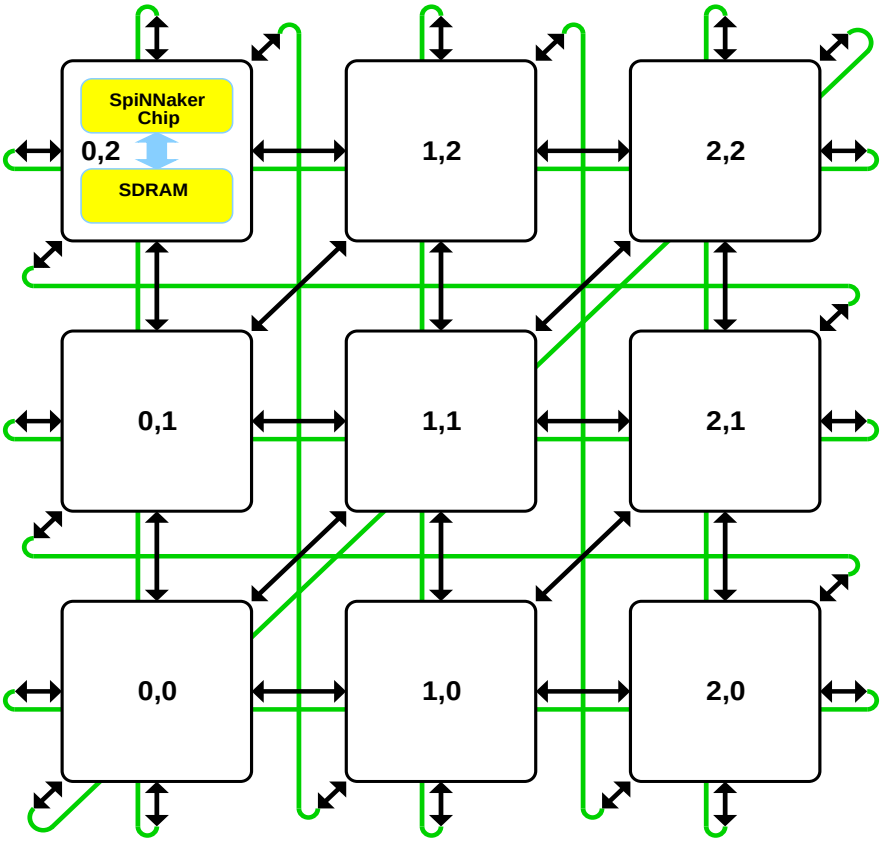
Certain registers in the System Controller have protection against corruption by errant code:



- Here any attempt to write the register must include the security code 0x5EC in the top 12 bits of the data word. If the security code is not present the write will have no effect.

2. System architecture

SpiNNaker is designed to form (with its associated SDRAM chip) a node of a massively parallel system. The system architecture is illustrated below:



2.1 Routing

The nodes are arranged in a *triangular* mesh with bidirectional links to *6* neighbours. The system supports multicast packets (to carry neural event information, routed by the associative Multicast Router), point-to-point packets (to carry system management and control information, routed by table look-up), nearest-neighbour packets (to support boot-time flood-fill and chip debug) and fixed-route packets (to convey application debug data back to the host computer).

Emergency routing

In the event of a link failing or congesting, traffic that would normally use that link is redirected in hardware around two adjacent links that form a triangle with the failed link. This “emergency routing” is intended to be temporary, and the operating system will identify a more permanent resolution of the problem. The local Monitor Processor is informed of uses of emergency routing.

Deadlock avoidance

The communications system has potential deadlock scenarios because of the possibility of circular dependencies between links. The policy used here to prevent deadlocks occurring is:

- ***no Router can ever be prevented from issuing its output***.

The mechanisms used to ensure this are:

- outputs have sufficient buffering and capacity detection so that the Router knows whether or not an output has the capacity to accept a packet;

- emergency routing is used, where possible, to avoid overloading a blocked output;
- where emergency routing fails (because, for example, the alternative output is also blocked) the packet is ‘dropped’ to a Router register, and the Monitor Processor informed;

The expectation is that the communications fabric will be lightly-loaded so that blocked links are very rare. Where the operating system detects that this is not the case it will take measures to correct the problem by modifying routing tables or migrating functionality.

Errant packet trap

Packets that get mis-routed could continue in the system for ever, following cyclic paths. To prevent this all (apart from nearest-neighbour) packets are time stamped and a coarse global time phase signal is used to trap old packets. To minimize overhead the time stamp is 2 bits, cycling 00 -> 01 -> 11 -> 10, and when the packet is two time phases old (time sent XOR time now = 0b11) it is dropped and an error flagged to the local Monitor Processor. The length of a time phase can be adapted dynamically to the state of the system; normally, timed-out packets should be very rare so the time phase can be conservatively long to minimise the risk of packets being dropped due to congestion.

2.2 Time references

A slow (nominally 32kHz) global reference clock is distributed throughout the system and is available to each processor via its DMA controller (which performs clock edge detection) and vectored interrupt controller. Software may use this to generate the local time phase information.

Each processor also has a timer/counter driven from the local processor clock which can be used to support time reference signals, for example a 1ms interrupt could be used to generate the time input to the real-time neural models.

2.3 System-level address spaces

The system incorporates different levels of component that must be enumerated:

- Each Node (where a Node is a SpiNNaker chip plus SDRAM) must have a unique, fixed address which is used as the destination ID for a point-to-point packet, and the addresses must be organised logically for algorithmic routing to function efficiently.
- Processors will be addressed relative to their host Node address, but this mapping will not be fixed as an individual Processor’s role can change over time. Point-to-point packets addressed to a Node will be delivered to the local Monitor Processor, whichever Processor is serving that function. Internal to a Node there is hard-wired addressing of each Processor for system diagnosis purposes, but this mapping will generally be hidden outside the Node.
- The neuron address space is purely a software issue and is discussed in ‘Application notes’ on page 95.

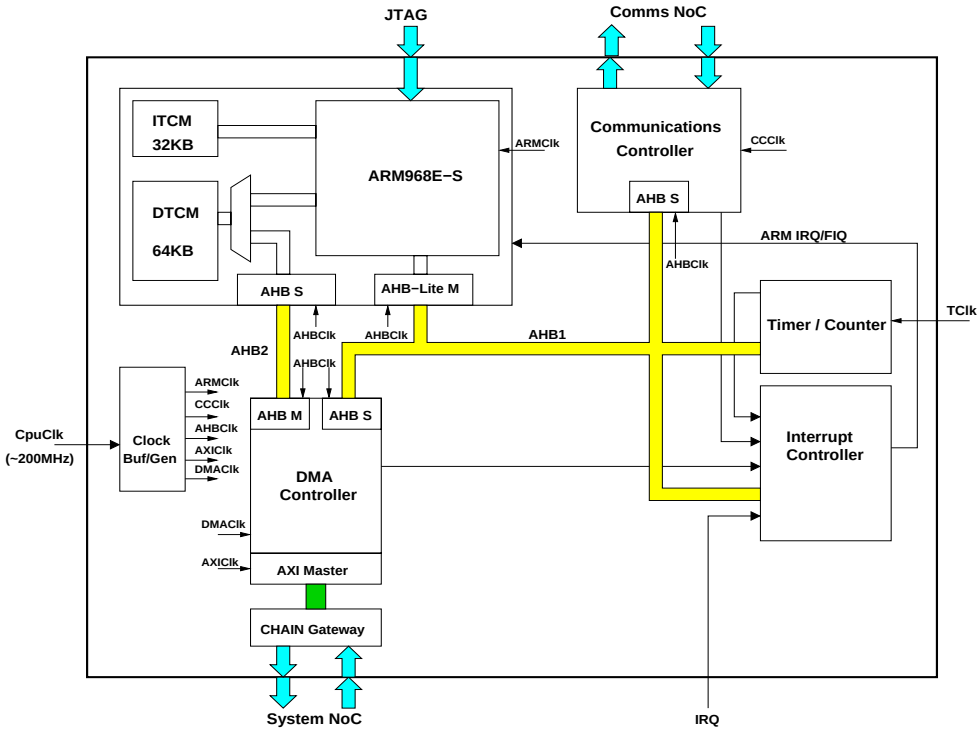
3. ARM968 processing subsystem

SpiNNaker incorporates *18* ARM968 processing subsystems which provide the computational capability of the device. Each of these subsystems is capable of generating and processing neural events communicated via the Communications NoC and, alternatively, of fulfilling the role of Monitor Processor.

3.1 Features

- a synthesized ARM968 module with:
  - an ARM9TDMI processor;
  - *32* Kbyte tightly-coupled instruction memory;
  - *64* Kbyte tightly-coupled data memory;
  - JTAG debug access.
- a local AHB with:
  - communications controller connected to Communications NoC;
  - DMA controller & interface to the System NoC;
  - timer/counter and interrupt controller.

3.2 ARM968 subsystem organisation



3.3 Memory Map

The memory map of the ARM968 spans a number of devices and buses. The tightly-coupled memories are directly connected to the processor and accessible at the processor clock speed. All other parts of the memory map are visible via the AHB master interface, which runs at the full processor clock rate. This gives direct access to the registers of the DMA controller, communications controller and the timer/interrupt controller. In addition, a path is available

through the DMA controller onto the System NoC which provides processor access to all memory resources on the System NoC. The memory map is defined as follows:

```
// ARM968 local memories
#define ITCM_START_ADDRESS          0x00000000 // instruction memory
#define DTCM_START_ADDRESS          0x00400000 // data memory

// Local peripherals - unbuffered write
#define COMM_CTL_START_ADDRESS_U    0x10000000 // Communications Controller
#define CTR_TIM_START_ADDRESS_U     0x11000000 // Counter-Timer
#define VIC_START_ADDRESS_U         0x1f000000 // vectored interrupt controller

// Local peripherals - buffered write
#define COMM_CTL_START_ADDRESS_B    0x20000000 // Communications Controller
#define CTR_TIM_START_ADDRESS_B     0x21000000 // Counter-Timer
#define VIC_START_ADDRESS_B         0x2f000000 // vectored interrupt controller

// DMA controller
#define DMA_CTL_START_ADDRESS_U     0x30000000 // DMA controller - unbuffered
#define DMA_CTL_START_ADDRESS_B     0x40000000 // DMA controller - buffered

// Unallocated; causes bus error    0x50000000 - 0x5fffffff

// SDRAM
#define SDRAM_START_ADDRESS_U       0x60000000 // SDRAM - buffered
#define SDRAM_START_ADDRESS_B       0x70000000 // SDRAM - unbuffered

// Unallocated; causes bus error    0x80000000 - 0xdfffffff

// System NoC peripherals - buffered write
#define PL340_APB_START_ADDRESS_B   0xe0000000 // PL340 APB port
#define RTR_CONFIG_START_ADDRESS_B  0xe1000000 // Router configuration
#define SYS_CTL_START_ADDRESS_B     0xe2000000 // System Controller
#define WATCHDOG_START_ADDRESS_B    0xe3000000 // Watchdog Timer
#define ETH_CTL_START_ADDRESS_B     0xe4000000 // Ethernet Controller
#define SYS_RAM_START_ADDRESS_B     0xe5000000 // System RAM
#define SYS_ROM_START_ADDRESS_B     0xe6000000 // System ROM

// Unallocated; causes bus error    0xe7000000 - 0xefffffff

// System NoC peripherals - unbuffered write
#define PL340_APB_START_ADDRESS_U   0xf0000000 // PL340 APB port
#define RTR_CONFIG_START_ADDRESS_U  0xf1000000 // Router configuration
#define SYS_CTL_START_ADDRESS_U     0xf2000000 // System Controller
#define WATCHDOG_START_ADDRESS_U    0xf3000000 // Watchdog Timer
#define ETH_CTL_START_ADDRESS_U     0xf4000000 // Ethernet Controller
#define SYS_RAM_START_ADDRESS_U     0xf5000000 // System RAM
#define SYS_ROM_START_ADDRESS_U     0xf6000000 // System ROM

// Unallocated; causes bus error    0xf7000000 - 0xfeffffff

// Boot area and VIC
#define BOOT_START_ADDRESS          0xff000000 // Boot area
#define HI_VECTORS                  0xffff0000 // high vectors (for boot)
#define VIC_START_ADDRESS_H         0xfffff000 // vectored interrupt controller
```

The areas shown against a yellow background are accessible only by their local ARM968 processor, not by a DMA controller nor by Nearest Neighbour packets via the Router (though of course the DMA controller can see the ITCM and DTCM areas through its second port, as these are the source/destination for DMA transfers). The DMA controller and Nearest Neighbour packets see the System RAM repeated across the bottom 16Mbytes of the address space from 0x00000000 to 0x00ffffff; the remainder of the yellow areas give undefined results and should not be addressed.

The ARM968 is configured to use high vectors after reset (to use the vectors in the Boot area), but then switched to low vectors once the ITCM is enabled and initialised.

The vectored interrupt controller (VIC) has to be at 0xfffff000 to enable efficient access to its vector registers.

All other peripherals start at a base address that can be formed with a single MOV immediate instruction.

4. ARM 968

The ARM968 (with its associated tightly-coupled instruction and data memories) forms the core processing resource in SpiNNaker.

4.1 Features

- ARM9TDMI processor supporting the ARMv5TE architecture.
- *32* Kbyte tightly-coupled instruction memory (I-RAM).
- *64* Kbyte tightly-coupled data memory (D-RAM).
- AHB interface to external system.
- JTAG-controlled debug access.
- support for Thumb and signal processing instructions.
- low-power halt and wait for interrupt function.

4.2 Organization

See ARM DDI 0311C – the ARM968E-S datasheet.

4.3 Fault-tolerance

Fault insertion

- ARM9TDMI can be disabled.
- Software can corrupt I-RAM and D-RAM to model soft errors.

Fault detection

- A chip-wide watchdog timer catches runaway software.
- Self-test routines, run at start-up and during normal operation, can detect faults.

Fault isolation

- The ARM968 unit can be disabled from the System Controller.
- Defective locations in the I-RAM and D-RAM can be mapped out of use by software.

Reconfiguration

- Software will avoid using defective I-RAM and D-RAM locations.
- Functionality will migrate to an alternative Processor in the case of permanent faults that go beyond the failure of one or two memory locations.

5. Vectored interrupt controller

Each processor node on an SpiNNaker chip has a vectored interrupt controller (VIC) that is used to enable and disable interrupts from various sources, and to wake the processor from sleep mode when required. The interrupt controller provides centralised management of IRQ and FIQ sources, and offers an efficient indication of the active sources for IRQ vectoring purposes.

The VIC is the ARM PL190, described in ARM DDI 0181E.

5.1 Features

- manages the various interrupt sources to each local processor.
- individual interrupt enables.
- routing to FIQ and/or IRQ,
  - there will normally be only one FIQ source: e.g. CC Rx ready, or a specific packet-type received.
- a central interrupt status view.
- a vector to the respective IRQ handler.
- programmable IRQ priority.
- interrupt sources:
  - Communication Controller flow-control interrupts;
  - DMA complete/error/timeout;
  - Timer 1 & 2 interrupts;
  - interrupt from another processor on the chip (usually the Monitor processor), set via a register in the System Controller;
  - packet-error interrupt from the Router;
  - system fault interrupt;
  - Ethernet controller;
  - off-chip signals;
  - 32kHz slow system clock;
  - software interrupt, for downgrading FIQ to IRQ.

5.2 Register summary

Base address: 0x2f000000 (buffered write), 0x1f000000 (unbuffered write), 0xfffff000 (high).

User registers

The following registers allow normal user programming of the VIC:

Name	Offset	R/W	Function
r0: VICirqStatus	0x00	R	IRQ status register
r1: VICfiqStatus	0x04	R	FIQ status register
r2: VICrawInt	0x08	R	raw interrupt status register
r3: VICintSel	0x0C	R/W	interrupt select register
r4: VICintEnable	0x10	R/W	interrupt enable register

Name	Offset	R/W	Function
r5: VICintEnClear	0x14	W	interrupt enable clear register
r6: VICsoftInt	0x18	R/W	soft interrupt register
r7: VICsoftIntClear	0x1C	W	soft interrupt clear register
r8: VICprotection	0x20	R/W	protection register
r9: VICvectAddr	0x30	R/W	vector address register
r10: VICdefVectAddr	0x34	R/W	default vector address register
VICvectAddr[15:0]	0x100-13c	R/W	vector address registers
VICvectCtrl[15:0]	0x200-23c	R/W	vector control registers

ID registers

In addition, there are test ID registers that will not normally be of interest to the programmer:

Name	Offset	R/W	Function
VICPeriphID0-3	0xFE0-C	R	Timer peripheral ID byte registers
VICPCID0-3	0xFF0-C	R	Timer Prime Cell ID byte registers

See the VIC Technical Reference Manual ARM DDI 0181E, for further details of the ID registers.

5.3 Register details

r0: IRQ status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ status																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

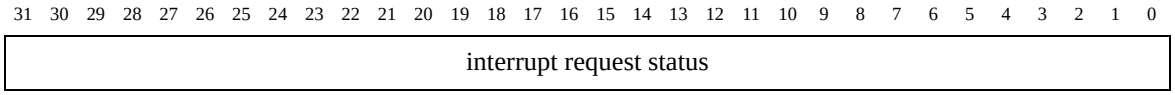
This read-only register yields the set of active IRQ requests (after masking).

r1: FIQ status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIQ status																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

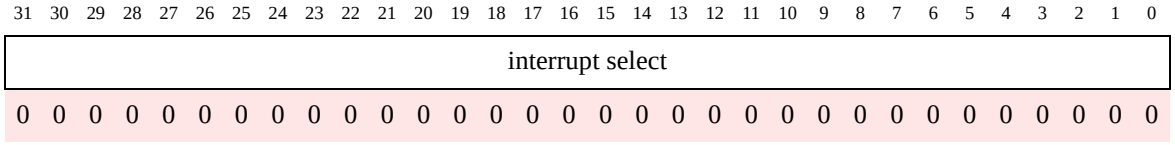
This read-only register yields the set of active FIQ requests (after masking).

r2: raw interrupt status



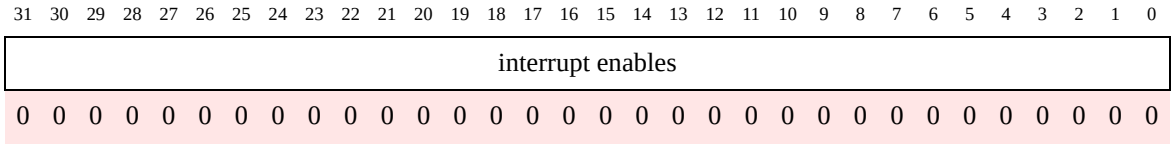
This read-only register yields the set of active input interrupt requests (before any masking).

r3: interrupt select



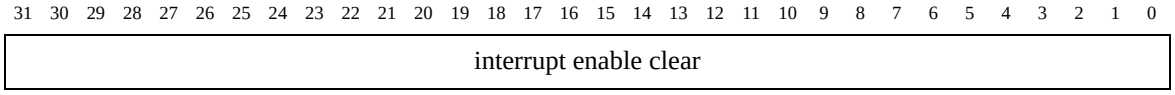
This register selects for each of the 32 interrupt inputs whether it gets sent to IRQ (0) or FIQ (1). The reset state is not specified (though is probably '0'?); all interrupts are disabled by r4 at reset.

r4: interrupt enable register



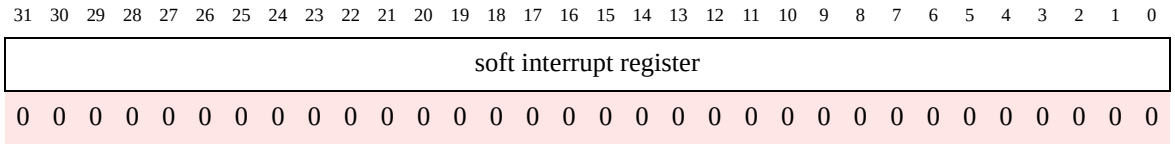
This register disables (0) or enables (1) each of the 32 interrupt inputs. Writing a '1' sets the corresponding bit in r4; writing a '0' has no effect. Interrupts are all disabled at reset.

r5: interrupt enable clear



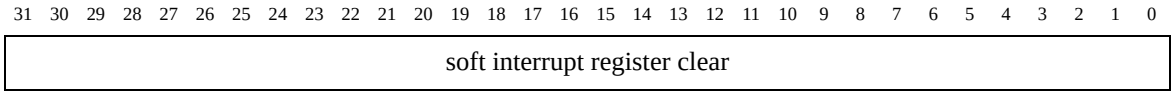
This write-only register selectively clears interrupt enable bits in r4. A '1' clears the corresponding bit in r4; a '0' has no effect.

r6: soft interrupt register



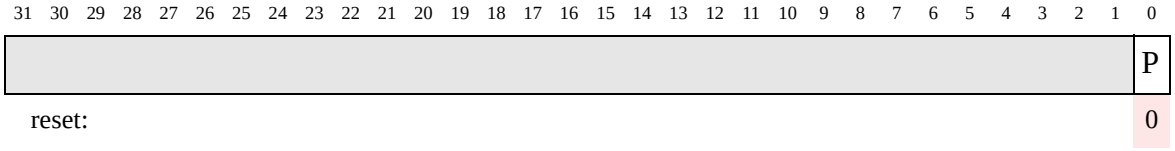
This register enables software to force interrupt inputs to appear high (before masking). A '1' written to any bit location will force the corresponding interrupt input to be active; writing a '0' has no effect. The reset state for these bits is unspecified, though probably '0'?

r7: soft interrupt register clear



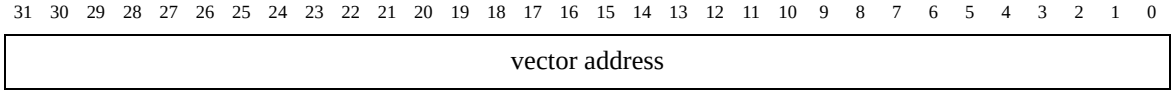
This write-only register selectively clears soft interrupt bits in r6. A ‘1’ clears the corresponding bit in r6; a ‘0’ has no effect.

r8: protection



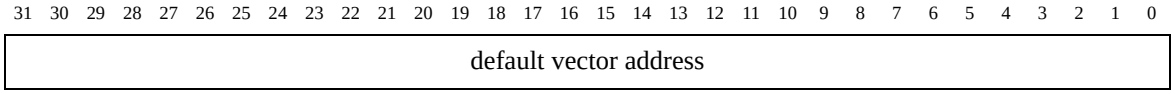
If the P bit is set VIC registers can only be accessed in a privileged mode; if it is clear then User-mode code can access the registers.

r9: vector address



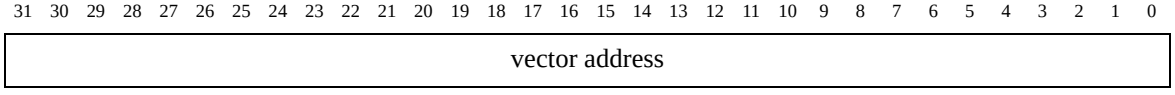
This register contains the address of the currently active interrupt service routine (ISR). It must be read at the start of the ISR, and written at the end of the ISR to signal that the priority logic should update to the next priority interrupt. Its state following reset is undefined.

r10: default vector address



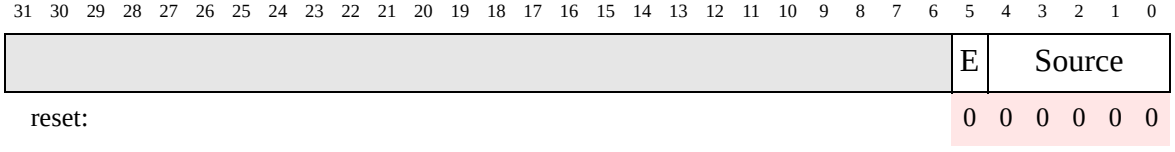
The default vector address is used by the 16 interrupts that are not vectored. Its state following reset is undefined.

vector address [15:0]



The vector address is the address of the ISR of the selected interrupt source. Their state following reset is undefined.

vector control [15:0]



The interrupt source is selected by bits[4:0], which choose one of the 32 interrupt inputs. The

interrupt can be enabled (E = 1) or disabled (E = 0). It is disabled following reset. The highest priority interrupt uses vector address [0] at offset 0x100 and vector control [0] at offset 0x200, and then successively reduced priority is given to vector addresses [1], [2], ... and vector controls [1], [2], ... at successively higher offset addresses.

5.4 Interrupt sources

19 of the 32 interrupt sources are local to the processor (and are coloured yellow in the table below) and 13 are from chip-wide sources (which will normally be enabled only in the Monitor Processor).

#	Name	Function
0	Watchdog	Watchdog timer interrupt
1	Software int	used only for local software interrupt generation
2	Comms Rx	the debug communications receiver interrupt
3	Comms Tx	the debug communications transmitter interrupt
4	Timer 1	Local counter/timer interrupt 1
5	Timer 2	Local counter/timer interrupt 2
6	CC Rx ready	Local comms controller packet received
7	CC Rx parity error	Local comms controller received packet parity error
8	CC Rx framing error	Local comms controller received packet framing error
9	CC Tx full	Local comms controller transmit buffer full
10	CC Tx overflow	Local comms controller transmit buffer overflow
11	CC Tx empty	Local comms controller transmit buffer empty
12	DMA done	Local DMA controller transfer complete
13	DMA error	Local DMA controller error
14	DMA timeout	Local DMA controller transfer timed out
15	Router diagnostics	Router diagnostic counter event has occurred
16	Router dump	Router packet dumped - indicates failed delivery
17	Router error	Router error - packet parity, framing, or time stamp error
18	Sys Ctl int	System Controller interrupt bit set for this processor
19	Ethernet Tx	Ethernet transmit frame interrupt
20	Ethernet Rx	Ethernet receive frame interrupt
21	Ethernet PHY	Ethernet PHY/external interrupt
22	Slow Timer	System-wide slow (nominally 32 KHz) timer interrupt
23	CC Tx not full	Local comms controller can accept new Tx packet
24	CC MC Rx int	Local comms controller multicast packet received

#	Name	Function
25	CC P2P Rx int	Local comms controller point-to-point packet received
26	CC NN Rx int	Local comms controller nearest neighbour packet received
27	CC FR Rx int	Local comms controller fixed route packet received
28	GPIO[0]	Signal on GPIO[0]
29	GPIO[1]	Signal on GPIO[1]
30	GPIO[6]	Signal on GPIO[6]
31	GPIO[7]	Signal on GPIO[7]

5.5 Fault-tolerance

Fault insertion

It is fairly easy to mess up vector locations, and to fake interrupt sources.

Fault detection

A failed vector location effectively causes a jump to a random location; this would be messy!

Fault isolation

Failed vector locations can be removed from service.

Reconfiguration

A failed vector location can be removed from service (provided there are enough vector locations available without it). Alternatively, the entire vector system could be shut down and interrupts run by software inspection of the IRQ and FIQ status registers.

6. Counter/timer

Each processor node on a SpiNNaker chip has a counter/timer.

The counter/timers use the standard AMBA peripheral device described on page 4-24 of the AMBA Design Kit Technical Reference Manual ARM DDI 0243A, February 2003. The peripheral has been modified only in that the APB interface of the original has been replaced by an AHB interface for direct connection to the ARM968 AHB bus.

6.1 Features

- the counter/timer unit provides two independent counters, for example for:
  - millisecond interrupts for real-time dynamics.
- free-running and periodic counting modes:
  - automatic reload for precise periodic timing;
  - one-shot and wrapping count modes.
- the counter clock (which runs at the processor clock frequency) may be pre-scaled by dividing by 1, 16 or 256.

6.2 Register summary

Base address: 0x21000000 (buffered write), 0x11000000 (unbuffered write).

User registers

The following registers allow normal user programming of the counter/timers:

Name	Offset	R/W	Function
r0: Timer1load	0x00	R/W	Load value for Timer 1
r1: Timer1value	0x04	R	Current value of Timer 1
r2: Timer1Ctl	0x08	R/W	Timer 1 control
r3: Timer1IntClr	0x0C	W	Timer 1 interrupt clear
r4: Timer1RIS	0x10	R	Timer 1 raw interrupt status
r5: Timer1MIS	0x14	R	Timer 1 masked interrupt status
r6: Timer1BGload	0x18	R/W	Background load value for Timer 1
r8: Timer2load	0x20	R/W	Load value for Timer 2
r9: Timer2value	0x24	R	Current value of Timer 2
r10: Timer2Ctl	0x28	R/W	Timer 2control
r11: Timer2IntClr	0x2C	W	Timer 2interrupt clear
r12: Timer2RIS	0x30	R	Timer 2raw interrupt status
r13: Timer2MIS	0x34	R	Timer 2masked interrupt status
r14: Timer2BGload	0x38	R/W	Background load value for Timer 2

Test and ID registers

In addition, there are test and ID registers that will not normally be of interest to the programmer:

Name	Offset	R/W	Function
TimerITCR	0xF00	R/W	Timer integration test control register
TimerITOP	0xF04	W	Timer integration test output set register
TimerPeriphID0-3	0xFE0-C	R	Timer peripheral ID byte registers
TimerPCID0-3	0xFF0-C	R	Timer Prime Cell ID byte registers

See AMBA Design Kit Technical Reference Manual ARM DDI 0243A, February 2003, for further details of the test and ID registers.

6.3 Register details

As both timers have the same register layout they can both be described as follows (X = 1 or 2):

r0/8: Timer X load value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load value for TimerX																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When written, the 32-bit value is loaded immediately into the counter, which then counts down from the loaded value. The background load value (r6/14) is an alternative view of this register which is loaded into the counter only when the counter next reaches zero.

r1/9: Current value of Timer X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TimerX current count																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This read-only register yields the current count value for Timer X.

r2/10: Timer X control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								E	M	I		Pre		S	O
reset:																								0	0	1		0	0	0	0

The shaded fields should be written as zero and are undefined on read. The functions of the remaining fields are described in the table below:

Name	bits	R/W	Function
E: Enable	7	R/W	enable counter/timer (1 = enabled)

Name	bits	R/W	Function
M: Mode	6	R/W	0 = free-running; 1 = periodic
I: Int enable	5	R/W	enable interrupt (1 = enabled)
Pre: TimerPre	3:2	R/W	divide input clock by 1 (00), 16 (01), 256 (10)
S: Timer size	1	R/W	0 = 16 bit, 1 = 32 bit
O: One shot	0	R/W	0 = wrapping mode, 1 = one shot

r3/11: Timer X interrupt clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Any write to this address will clear the interrupt request.

r4/12: Timer X raw interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															R
reset:																															0

Bit zero yields the raw (unmasked) interrupt request status of this counter/timer.

r5/13: Timer X masked interrupt status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															M
reset:																															0

Bit zero yields the masked interrupt status of this counter/timer.

r6/14: Timer X background load value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Background load value for TimerX																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The 32-bit value written to this register will be loaded into the counter when it next counts down to zero. Reading this register will yield the same value as reading register 0/8.

6.4 Fault-tolerance

Fault insertion

Disabling a counter (by clearing the E bit in its control register) will cause it to fail in its function.

Fault detection

Use the second counter/timer with a longer period to check the calibration of the first?

Fault isolation

Disable the counter/timer with the E bit in the control register; disable its interrupt output; disable the interrupt in the interrupt controller.

Reconfiguration

If one counter fails then a system that requires only one counter can use the other one.

7. DMA controller

Each ARM968 processing subsystem includes a DMA controller. The DMA controller is primarily used for transferring inter-neural connection data from the SDRAM in large blocks in response to an input event arriving at a fascicle processor, and for returning updated connection data during learning. In addition, the DMA controller can transfer data to/from other targets on the System NoC such as the System RAM and Boot ROM.

As a secondary function the DMA controller incorporates a 'Bridge' across which its host ARM968 has direct read and write access to System NoC devices, including the SDRAM. The ARM968 can use the Bridge whether or not DMA transfers are active.

7.1 Features

- DMA engine supporting parallel operations:
  - DMA transfers;
  - direct pass-through requests from the ARM968;
  - dual buffers supporting simultaneous direct and DMA transfers.
- Support for CRC error control in transferred blocks.
- Interrupt-driven or polled DMA completion notification:
  - DMA complete interrupt signal;
  - various DMA error interrupt signals;
  - DMA time-out interrupt signal.
- Parameterisable buffer sizes.
- Direct and DMA request queueing.

7.2 Using the DMA controller

There are 2 types of requests for DMA controller services. DMA transfers are initiated by writing to control registers in the controller, executed in the background, and signal an interrupt when complete. Bridge transfers occur when the ARM initiates a request directly to the needed device or service. The DMA controller fulfills these requests transparently, the host processor retaining full control of the transfer. Invisible to the user, the controller may buffer the data from write requests for more efficient bus management. If an error occurs on such a buffered write the DMA controller can signal an error interrupt.

The controller acts as a Bridge between the AHB bus on the ARM AHB slave interface and the AXI interface on the system NoC, performing the required address and control resequencing (stripping addresses from non-first beats of a burst), data flow management and request arbitration. The arbiter prioritises requests in the following order:

1. Bridge reads,
2. Bridge writes,
3. DMA burst requests.

No request can gain access to the AXI interface until any active burst transaction on the interface has completed. Read requests while a DMA transfer is in progress require special handling. The read must wait until any active request has completed, and therefore a Bridge read could stall the processor and AHB slave bus for many cycles. In addition, if buffered writes exist, potential data coherency conflicts exist. The recommended procedure is for the ARM processor to interrogate the WB active (A) bit in the DMA Status register (STAT) before requesting a Bridge read.

To initiate a DMA transfer, the ARM must write to the following registers in the DMA controller: System Address (ADRS), TCM Address (ADRT), and Description (DESC). The order of writing of

the first two register operations is not important, but the Description write must be the last as it commits the DMA transfer. The processor may also optionally write the CRC and Global Control (GCTL) registers to set up additional parameters. The expected model, however, is that these registers are updated infrequently, perhaps only once after power-up. The processor may read from any register at any time. The processor may have a maximum of 2 submitted DMA requests of which only one will be active. When the transfer queue is empty (as indicated by the Q bit in the Status (STAT) register), the processor may queue another request.

Accesses to DMA Controller registers are restricted to programs running on the ARM968 in privileged (i.e. non-user) modes. Attempts to access these registers in user mode will result in a bus error.

An attempt to write register r1 to r3 when the queue is full will result in a bus error.

Any access (read or write) to a non-existent register will result in a bus error.

Non-word-aligned addresses and byte and half-word accesses will result in a bus error.

7.3 Register summary

Base address: 0x40000000 (buffered write), 0x30000000 (unbuffered write).

Name	Offset	R/W	Function
r0: unused	0x00		
r1: ADRS	0x04	R/W	DMA address on the system interface
r2: ADRT	0x08	R/W	DMA address on the TCM interface
r3: DESC	0x0C	R/W	DMA transfer description
r4: CTRL	0x10	R/W	Control DMA transfer
r5: STAT	0x14	R	Status of DMA and other transfers
r6: GCTL	0x18	R/W	Control of the DMA device
r7: CRCC	0x1C	R	CRC value calculated by CRC block
r8: CRCR	0x20	R	CRC value in received block
r9: TMTV	0x24	R/W	Timeout value
r10: StatsCtl	0x28	R/W	Statistics counters control
r16-23: Stats0-7	0x40-5C	R	Statistics counters
r64: unused	0x100		
r65: AD2S	0x104	R	Active system address
r66: AD2T	0x108	R	Active TCM address
r67: DES2	0x10C	R	Active transfer description
r96-r127	0x180-1FC	R/W	CRC polynomial matrix

7.4 Register details

r0: unused

r1: ADRS - System Address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
System Address																														00	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The 32-bit start byte address on the system interface. Note that a read is considered a data movement from a source on the system bus to a destination on the TCM bus. DMA transfers are word-aligned, so bits[1:0] are fixed at zero.

r2: ADRT - TCM Address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCM Address																														00	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The 32-bit start address on the TCM interface.

r3: DESC - DMA transfer description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Transfer ID						P	W	burst			C	D			Length															00	
0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The function of these fields is described in the table below:

Name	bits	R/W	Function
Transfer ID	31:26	R/W	software defined transfer ID
P: Privilege	25	R/W	DMA transfer mode is user (0) or privileged (1)
W: Width	24	R/W	transfer width is word (0) or double-word (1)
Burst	23:21	R/W	burst length = 2^B x Width, B = 0..4 (i.e max 16)
C: CRC	20	R/W	check (read) or generate (write) CRC
D: Direction	19	R/W	read from (0) or write to (1) system bus
Length	16:2	R/W	length of the DMA transfer, in words

The TCM as currently implemented has a size of 64Kbytes (for the data TCM). A DMA transfer must of necessity either take as a source or a destination the TCM, justifying this restriction. DMA transfers are word-aligned, so bits[1:0] are fixed at zero.

The Burst length defines the unit of transfer (in words or double-words, depending on W) across the

System NoC. Longer bursts will in general make more efficient use of the available SDRAM bandwidth.

Note that the Length **excludes** the 32-bit CRC word, if CRC is used.

Writing to this register automatically commits a transfer as defined by the values in r1-r3.

r4: CTRL - Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										W	T	D	R	A	U
reset:																										0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
W: clear WB Int	5	R/W	clear Write Buffer interrupt request
T: clear Timeout Int	4	R/W	clear Timeout interrupt request
D: clear Done Int	3	R/W	clear Done interrupt request
R: Restart	2	R/W	resume transfer (clears DMA errors)
A: Abort	1	R/W	end current transfer and discard data
U: Uncommit	0	R/W	setting this bit uncommits a queued transfer

These bits can only be set to 1 by the user, they cannot be reset. Writing a 0 has no effect. They will clear automatically once they have taken effect, which will be at the next safe opportunity, typically between transfer bursts.

r5: STAT - Status Register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
processor ID											Condition Codes																A	F	Q	P	T	
hardwired proc ID											0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	

The functions of these fields are described in the table below:

Name	bits	R/W	Function
processor ID	31:24	R	hardwired processor ID identifies CPU on chip
Condition Codes	20:10	R	DMA condition codes
A: WB active	4	R	write buffer is not empty
F: WB full	3	R	write buffer is full
Q: Queue full	2	R	DMA transfer is queued - registers are full
P: Paused	1	R	DMA transfer is PAUSED

Name	bits	R/W	Function
T: Transferring	0	R	DMA transfer in progress

The condition codes are defined as follows:

Name	bits	R/W	Function
Write buff error	20	R	a buffered write transfer has failed
TBD	19:18	R	not yet allocated
Soft reset	17	R	a soft reset of the DMA controller has happened
User abort	16	R	the user has aborted the transfer (via r4)
AXI error	15	R	the AXI interface has signalled a transfer error
TCM error	14	R	the TCM AHB interface has signalled an error
CRC error	13	R	the calculated and received CRCs differ
Timeout	12	R	a burst transfer has not completed in time
2nd transfer done	11	R	2nd DMA transfer has completed without error
Transfer done	10	R	a DMA transfer has completed without error

When a DMA error occurs the corresponding condition code flag is set, the DMA engine is PAUSED (bit[1]) and the current transfer is terminated. A queued transfer remains in the queue but is not started. A new transfer can be committed if the queue is empty, but it will not start until the DMA controller is brought out of PAUSE. AD2S, AD2T and DES2 (r65-67) contain information about the failed transfer and can be used to diagnose the problem. A restart command (r4 bit[2]) is required to bring the DMA controller out of PAUSE. This will clear the error codes [16:13] and restart DMA operation. The terminated transfer must be restarted explicitly by software if this is required.

A soft reset will set bit[17], clear the transfer queue and take the DMA controller into the IDLE state. The DMA controller is not PAUSED, and new transfers can be committed and start immediately. A restart command (r4 bit[2]) is required to clear the soft reset flag [17] - starting a new transfer does NOT clear it.

Timeout [12] and Write Buffer error [20] have explicit clears in CTRL.

The two transfer done bits [11:10] count up through the sequence 00 -> 01 -> 11 as DMA transfers complete, and count down through the reverse sequence when a 1 is written to CTRL[3]. As a result of this coding, Transfer Done [10] can be read as indicating that at least one DMA transfer has completed, and a second completed transfer can be handled by inspecting bit[11] in software or left to be handled by a subsequent Transfer Done interrupt.

r6: GCTL - Global Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T											Interrupt enables																				B
0											0	0	0	0	0	0	0	0	0	0	0										0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
T: Timer	31	R/W	system-wide slow timer status and clear
Interrupt enables	20:10	R/W	respective interrupt enables for the r5 conditions
B: Bridge buffer	0	R/W	enable Bridge write buffer

The DMA controller passes four interrupt request lines to the VIC:

- dmac_done: the logical OR of GCTL[11:10] & STAT[11:10]
- dmac_timeout: GCTL[12] & STAT[12]
- dmac_error: the logical OR of GCTL[20:13] & STAT[20:13]
- system-wide slow (nominally 32 KHz) timer interrupt

Note that write buffer errors and timeout errors do NOT stop the DMA engine nor the transfer in progress.

The system-wide slow timer is a clock signal that sets bit[31] on every rising edge, thereby raising an interrupt request to the VIC, and is cleared by writing a 0 to bit[31]. Writing a 1 to bit[31] has no effect.

r7: CRCC - Calculated CRC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_value (calculated)																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This is the 32-bit CRC value calculated by the DMA CRC unit.

r8: CRCR - Received CRC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_value (received)																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This is the 32-bit CRC value read in the block of data loaded by a DMA transfer.

r9: TMTV - Timeout value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						V					00000				
reset:																						0	0	0	0	0	0	0	0	0	0

This is a 10-bit counter value used to determine when the DMA controller should timeout on an attempted transfer burst. The count units are clock cycles. When TMTV = 0 the timeout counter is disabled. Note that a timeout will not stop the transfer.

r10: StatsCtl - Statistics counters control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														C	E
reset:																														0	0

E, bit[0], enables the statistics counters (r16-23).

Writing ‘1’ to C, bit[1], zeroes the statistics counters. Writing a ‘0’ has no effect. Bit[1] always reads ‘0’.

r16-23: Stats0-7 - Statistics counters

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																Count0-7															
reset:																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

These eight 16-bit counter registers record statistics relating to the latency of DMA transations across the System NoC. Count0 records the number of transactions that complete in 0-127 clock cycles, Count1 128-255 clock cycles, and so on up to Count7 which counts transactions that complete in 896+ clock cycles.

The counters are enabled and cleared via r10.

r65-67: Active DMA transfer registers

These registers are not directly written. They reflect the state of the active DMA transfer, with AD2S and AD2T holding the respective System and TCM addresses to be used in the next burst of the transfer, and DES2 holding the description of the transfer in progress (the remaining length, ID, burst size, and direction).

r96-127: CRC polynomial matrix

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_polynomial row[31:0]																															

The CRC hardware is highly programmable and can be used in a number of ways to detect, and possibly correct, errors in blocks of data transferred by the DMA controller between the ARM968 DTCM and the off-chip SDRAM.

For example, to use the Ethernet 32-bit CRC with polynomial 0x04C11DB7, the following 32 hexadecimal values should be programmed into r96-127:

```
FB808B20, 7DC04590, BEE022C8, 5F701164, 2FB808B2, 97DC0459, B06E890C, 58374486,
AC1BA243, AD8D5A01, AD462620, 56A31310, 2B518988, 95A8C4C4, CAD46262, 656A3131,
493593B8, 249AC9DC, 924D64EE, C926B277, 9F13D21B, B409622D, 21843A36, 90C21D1B,
33E185AD, 627049F6, 313824FB, E31C995D, 8A0EC78E, C50763C7, 19033AC3, F7011641.
```

The CRC unit is configurable to use a different 32-bit polynomial, a different polynomial length, and a different data word length. For example, it can be configured to compute CRC16 separately for each half-word of the data stream. A Matlab program can be used to determine the appropriate polynomial matrix values.

7.5 Fault-tolerance

Fault insertion

Software can introduce errors in data blocks in SDRAM which should be trapped by the CRC hardware.

Fault detection

The CRC unit can detect errors in the data transferred by the DMA controller.

The DMA controller will time-out if a transaction takes too long.

Fault isolation

The DMA Controller is mission-critical to the local processing subsystem, so if it fails the subsystem should be disabled and isolated.

Reconfiguration

The local processing subsystem is shut down and its functions migrated to another subsystem on this or another chip. It should be possible to recover all of the subsystem state and to migrate it, via the SDRAM, to a functional alternative.

8. Communications controller

Each processor node on SpiNNaker includes a communications controller which is responsible for generating and receiving packets to and from the communications network.

8.1 Features

- Support for 4 packet types:
  - multicast (MC) neural event packets routed by a key provided at the source;
  - point-to-point (P2P) packets routed by destination address;
  - nearest-neighbour (NN) packets routed by arrival port;
  - fixed-route (FR) packets routed by the contents of a register.
- Packets are either 40 or 72 bits long. The longer packets carry a 32-bit payload.
- 2-bit time stamp (used by Routers to trap errant packets).
- Parity (to detect some corrupt packets).

8.2 Packet formats

Neural event multicast (MC) packets (type 0)

Neural event packets include a control byte and a 32-bit routing key inserted by the source. In addition they may include an optional 32-bit payload:

8 bits	32 bits	32 bits
control	routing key	optional payload

The 8-bit control field includes packet type (bits[7:6] = 00 for multicast packets), emergency routing and time stamp information, a payload indicator, and error detection (parity) information:

7	6	5	4	3	2	1	0
0	0	emergency routing		time stamp		payload	parity

Point-to-point (P2P) packets (type 1)

Point-to-point packets include 16-bit source and destination chip IDs, plus a control byte and an optional 32-bit payload:

8 bits	16 bits	16 bits	32 bits
control	source ID	destination ID	optional payload

Here the 8-bit control field includes packet type (bits[7:6] = 01 for P2P packets), a sequence code, time stamp, a payload indicator and error detection (parity) information:

7	6	5	4	3	2	1	0
0	1	seq code		time stamp		payload	parity

Nearest-neighbour (NN) packets (type 2)

Nearest-neighbour packets include a 32-bit address or operation field, plus a control byte and an optional 32-bit payload:

8 bits	32 bits	32 bits
control	address/operation	optional payload

Here the 8-bit control field includes packet type (bits[7:6] = 10 for NN packets), a 'peek/poke' or 'normal' type indicator (T), routing information, a payload indicator and error detection (parity) information:

7	6	5	4	3	2	1	0
1	0	T	route			payload	parity

Fixed-Route (FR) packets (type 3)

Fixed-route packets include a 32-bit payload field, plus a control byte and an optional 32-bit payload extension:

8 bits	32 bits	32 bits
control	payload	optional payload extension

Here the 8-bit control field includes packet type (bits[7:6] = 11 for FR packets), emergency routing and time stamp information, a payload indicator, and error detection (parity) information:

7	6	5	4	3	2	1	0
1	1	emergency routing		time stamp		payload	parity

8.3 Control byte summary

The various fields in the control bytes of the different packet types are summarised below:

Field Name	bits	Function
parity	0	parity of complete packet (including payload when used)
payload	1	data payload (1) or no data payload (0)
time stamp	3:2	phase marker indicating time packet was launched
seq code	5:4	P2P only: sequence code, software defined
emergency routing	5:4	MC & FR: used to control routing around a failed link
route	4:2	NN only: information for the Router
T: NN packet type	5	NN only: packet type - normal (0) or peek/poke (1)
packet type	7:6	= 00 for MC; = 01 for P2P; = 10 for NN; = 11 for FR

parity

The complete packet (including the data payload where used) will have odd parity.

data

Indicates whether the packet has a 32-bit data payload (= 1) or not (= 0).

time stamp

The system has a global time phase that cycles through 00 -> 01 -> 11 -> 10 -> 00. Global synchronisation must be accurate to within one time phase (the duration of which is programmable and may be dynamically variable). A packet is launched with a time stamp equal to the current time phase, and if a Router finds a packet that is two time phases old (time now XOR time launched = 11) it will drop it to the local Monitor Processor. The time stamp is inserted by the local Router if the route field in SAR (see ‘Register details’ on page 33) is 111, which is the normal case, so the Communication Controller need do nothing here. If SAR holds a different value in the route field the time stamp from TCR is used.

seq code

P2P packets may use these bits (under software control) to indicate the sequence of data payloads, or for other purposes.

emergency routing

MC & FR packets use these bits to control emergency routing around a failed or congested link:

- 00 -> normal packet;
- 01 -> the packet has been redirected by the previous Router through an emergency route along with a normal copy of the packet. The receiving Router should treat this as a combined normal plus emergency packet.
- 10 -> the packet has been redirected by the previous Router through an emergency route which would not be used for a normal packet.
- 11 -> this emergency packet is reverting to its normal route.

route

These bits are set at packet launch to the values defined in the control register. They enable a packet to be directed to a particular neighbour (0 - 5), broadcast to all or a subset (as defined in the Router r33 ‘NN broadcast’ bits - see ‘r33: fixed-route packet routing’ on page 49) of neighbours (6), or to the local Monitor Processor (7).

T (NN packet type)

This bit specifies whether an NN packet is ‘normal’, so that it is delivered to the Monitor Processor on the neighbouring chip(s), or ‘peek/poke’, so that performs a read or write access to the neighbouring chip’s System NoC resource.

packet type

These bits indicate whether the packet is a multicast (00), point-to-point (01), nearest-neighbour (10) or fixed-route (11) packet.

8.4 Debug access to neighbouring devices

The ‘peek’ and ‘poke’ mechanism gives access to the System NoC address space on any neighbouring device without processor intervention on that chip. To read a word, include its address in a ‘peek/poke’ nearest neighbour packet output (i.e. with the T bit set). Only word addresses are permitted. The absence of a payload indicates that a read (‘peek’) is required. This would normally be done by a Monitor Processor although, in principle, any processor can output

this packet.

The target device performs the appropriate access and returns a response on the corresponding link input. This is delivered to the processor designated as Monitor Processor in the local router. The response is a 'normal' NN packet which carries the requested word as payload. The address field is also returned for identification purposes with the least significant bit set to indicate a response. Bit 1 of the address will also be set if the access caused a bus error.

Writing ('poke') is similar; including a payload in the outgoing packet causes that word to be written. A payload-less response packet is returned which will indicate the error status.

8.5 Register summary

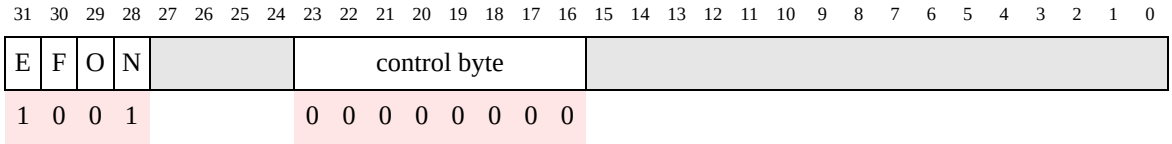
Base address: 0x20000000 (buffered write), 0x10000000 (unbuffered write).

Name	Offset	R/W	Function
r0: TCR (Tx control)	0x00	R/W	Controls packet transmission
r1: TDR (Tx data)	0x04	W	32-bit data for transmission
r2: TKR (Tx key)	0x08	W	Send MC key/P2P dest ID & seq code
r3: RSR (Rx status)	0x0C	R/W	Indicates packet reception status
r4: RDR (Rx data)	0x10	R	32-bit received data
r5: RKR (Rx key)	0x14	R	Received MC key/P2P source ID & seq code
r6: SAR (Source addr)	0x18	R/W	P2P source address
r7: TSTR (test)	0x1C	R/W	Used for test purposes

A packet will contain a data payload if r1 is written before r2; this can be performed using an ARM STM instruction.

8.6 Register details

r0: TCR - transmit control



The functions of these fields are described in the table below:

Name	bits	R/W	Function
E: empty	31	R	Tx buffer empty
F: full	30	R/W	Tx buffer full (sticky)
O: overrun	29	R/W	Tx buffer overrun (sticky)
N: not full	28	R	Tx buffer not full, so it is safe to send a packet

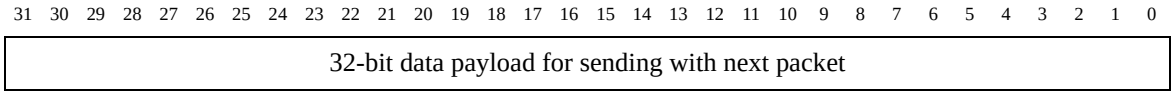
Name	bits	R/W	Function
control byte	23:16	W	control byte of next sent packet

The parity field in the control byte will be replaced by an automatically-generated value when the packet is launched, and the sequence field will be replaced by the value in TKR. The time stamp (where applicable) will be inserted by the local Router if the route field in SAR is 111, otherwise the value here will be used.

The transmit buffer full and not full controls are expected to be used, by polling or interrupt, to prevent buffer overrun. Tx buffer full is sticky and, once set, will remain set until 0 is written to bit 30. Transmit buffer overrun indicates packet loss and will remain set until explicitly cleared by writing 0 to bit 29.

E, F, O and N reflect the levels on the Tx interrupt signals sent to the interrupt controller.

r1: TDR - transmit data payload

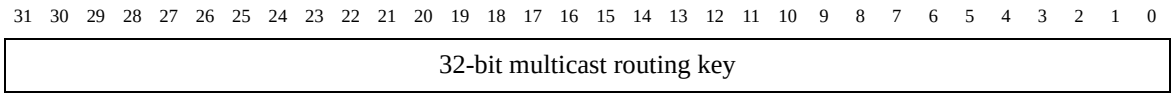


If data is written into TDR before a send key or dest ID is written into TKR, the packet initiated by writing to TKR will include the contents of TDR as its data payload. If no data is written into TDR before a send key or dest ID is written into TKR the packet will carry no data payload.

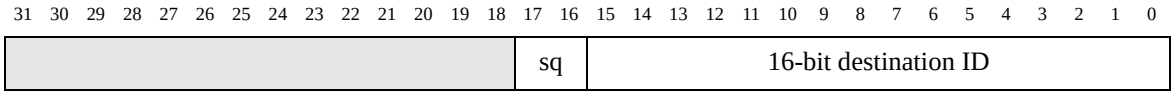
r2: TKR - send MC key or P2P dest ID & sequence code

Writing to TKR causes a packet to be issued (with a data payload if TDR was written previously).

If bits[23:22] of the control register in TCR are 00 the Communication Controller is set to send multicast packets and a 32-bit routing key should be written into TKR. The 32-bit routing key is used by the associative multicast Routers to deliver the packet to the appropriate destination(s).



If bits[23:22] of the control register are 01 the Communication Controller is set to send point-to-point packets and the value written into TKR should include the 16-bit address of the destination chip in bits[15:0] and a sequence code in bits[17:16]. (See ‘seq code’ on page 32.)



If bits[23:22] of the control register are 10 the Communication Controller is set to send nearest neighbour packets and the 32-bit NN address/operation field should be written in TKR.

If bits[23:22] of the control register are 11 the Communications Controller is set to send fixed-route packets and the value written into TKR is a 32-bit payload, possibly augmented by a further 32 bits in TDR if this was written previously.

r3: RSR - receive status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T	A	E		Route			control byte																				F	N	P	M
0	0	0	0		0	0	0	0	0	0	0	0	0	0	0													0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
R: received	31	R	Rx packet received
T: parity	30	R/W	Rx packet parity error (sticky)
A: framing error	29	R/W	Rx packet framing error (sticky)
E: error-free	28	R	Rx packet received without error
Route	26:24	R	Rx route field from packet
Control byte	23:16	R	Control byte of last Rx packet
F: FR packet	3	R	error-free fixed-route packet received
N: NN packet	2	R	error-free nearest-neighbour packet received
P: P2P packet	1	R	error-free point-to-point packet received
M: MC packet	0	R	error-free multicast packet received

Any packet that is received will set R, which will remain set until RKR has been read. A packet that is received with a parity and/or framing error also sets T and/or A. These bits remain set until explicitly reset by writing 0 to bit 30 or bit 29 respectively.

R, T, A, M, P, N & F reflect the levels on the Rx interrupt signals sent to the interrupt controller.

Note that these status bits will have a one-cycle latency before becoming valid so, for example, checking R one cycle after reading RKR will return 1, the old value.

r4: RDR - received data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit received data payload																															

If a received packet carries a data payload the payload will be delivered here and will remain valid until r5 is read.

r5: RKR - received MC key or P2P source ID & sequence code

A received packet will deliver its MC routing key, NN address or P2P source ID and sequence code to RKR. For an MC or NN packet this will be the exact value that the sender placed into its TKR for transmission; for a P2P packet the sequence number will be that placed by the sender into its TKR, and the 16-bit source ID will be that in the sender's SAR.

The register is read sensitive - once read it will change as soon as the next packet arrives.

r6: SAR - source address and route

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Route											p2p source ID															
reset:					1	1	1									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
Route	26:24	W	Set ‘fake’ route in packet
P2P source ID	15:0	W	16-bit chip source ID for P2P packets

The P2P source ID is expected to be configured once at start-up.

The route field allows a packet to be sent by a processor to the router which appears to have come from one of the external links. Normally this field will be set to 7 (0b111) but can be set to a link number in the range 0 to 5 to achieve this.

r7: TSTR - test

Setting bit 0 of this register makes all registers read/write for test purposes. Clearing bit 0 restricts write access to those register bits marked as read-only in this datasheet. All register bits may be read at any time. Bit 0 is cleared by reset.

8.7 Fault-tolerance

Fault insertion

Software can cause the Communications Controller to misbehave in several ways including inserting dodgy routing keys, source IDs, destination IDs.

Fault detection

Parity of received packet; received packet framing error; transmit buffer overrun.

Fault isolation

The Communications Controller is mission-critical to the local processing subsystem, so if it fails the subsystem should be disabled and isolated.

Reconfiguration

The local processing subsystem is shut down and its functions migrated to another subsystem on this or another chip. It should be possible to recover all of the subsystem state and to migrate it, via the SDRAM, to a functional alternative.

9. Communications NoC

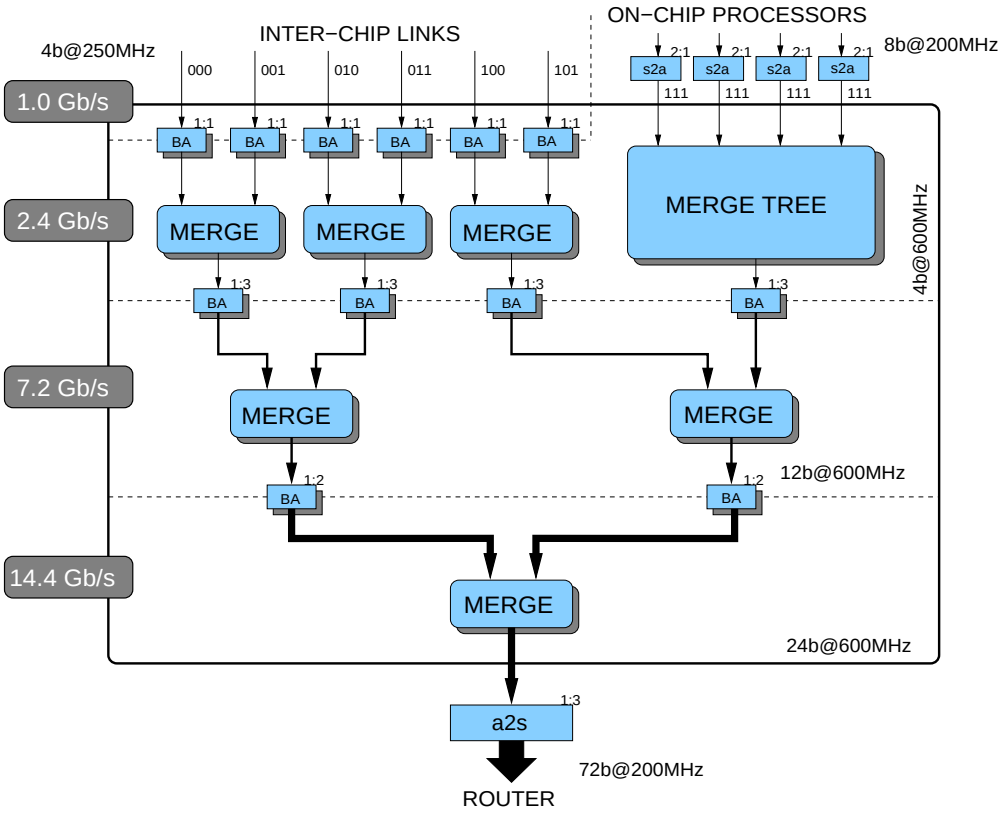
The Communications NoC carries packets between the processors on the same or different chips. It plays a central role in the system architecture. Its connectivity to the other components is shown the the chip block diagram in ‘Chip organization’ on page 5.

9.1 Features

- On- and inter-chip links
- Router which handles multicast, point-to-point, nearest neighbour and fixed-route packets.
- Arbiter to merge all sources into a sequential packet stream into the Router.
- Individual links can be reset to clear blockages and deadlocks.

9.2 Input structure

The input structure is a tree Arbiter which merges the various sources of packets into a single stream. Its structure is illustrated below. The numbers indicate source tagging of the packets.



9.3 Output structure

The Router produces separate outputs to all on-chip processor nodes and to the off-chip links, so the output connectivity is a set of individual self-timed links.

10. Router

The Router is responsible for routing all packets that arrive at its input to one or more of its outputs. It is responsible for routing multicast neural event packets, which it does through an associative multicast router subsystem, point-to-point packets (for which it uses a look-up table), nearest-neighbour packets (using a simple algorithmic process), fixed-route packet routing (defined in a register), default routing (when a multicast packet does not match any entry in the multicast router) and emergency routing (when an output link is blocked due to congestion or hardware failure).

Various error conditions are identified and handled by the Router, for example packet parity errors, time-out, and output link failure.

10.1 Features

- *1,024* programmable associative multicast (MC) routing entries.
  - associative routing based on source 'key';
  - with flexible 'don't care' masking;
- look-up table routing of point-to-point (P2P) packets.
- routing of nearest-neighbour (NN) and fixed-route (FR) packets.
- support for 40- and 72-bit packets.
- default routing of unmatched multicast packets.
- automatic 'emergency' re-routing around failed links.
  - programmable wait time before emergency routing and before dropping packet.
- pipelined implementation to route 1 packet per cycle (peak).
  - back-pressure flow control;
  - power-saving pipeline control.
- failure detection and handling:
  - packet parity error;
  - time-expired packet;
  - output link failure;
  - packet framing (wrong length) error.

10.2 Description

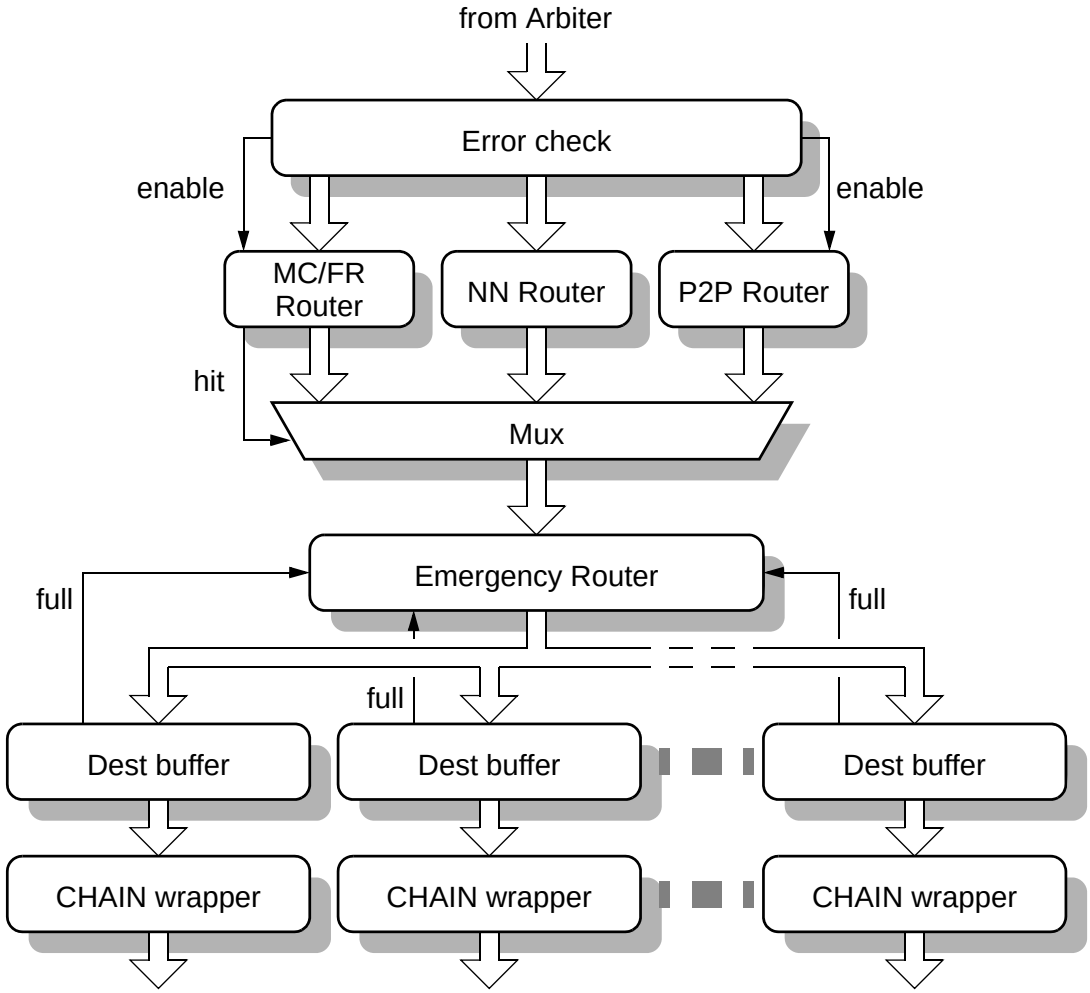
Packets arrive from other nodes via the link receiver interfaces and from internal processor nodes and are presented to the router one-at-a-time. The Arbiter is responsible for determining the order of presentation of the packets, but as each packet is handled independently the order is unimportant (though it is desirable for packets following the same route to stay in order).

Each multicast packet contains an identifier that is used by the Router to determine which of the outputs the packet is sent to. These outputs may include any subset of the output links, where the packet may be sent via the respective link transmitter interface, and/or any subset of the internal processor nodes, where the packet is sent to the respective Communications Controller.

For the neural network application the identifier can be simply a number that uniquely identifies the source of the packet – the neuron that generated the packet by firing. This is 'source address routing'. In this case the packet need contain only this identifier, as a neural spike is an 'event' where the only information is that the neuron has fired. The Router then functions simply as a look-up table where for each identifier it looks up a routing word, where each routing word contains 1 bit for each destination (each link transmitter interface and each local processor) to indicate whether or not the packet should be passed to that destination.

10.3 Internal organization

The internal organization of the Router is illustrated in the figure below.



Packets are passed as complete 40- or 72-bit units from the Arbiter, together with the identity of the Rx interface that the packet arrived through (for nearest-neighbour, emergency and default routing). The first stage of processing here is to identify errors. The second stage passes the packet to the appropriate routing engines – the multicast (MC) router is activated only if the packet is error-free and of multicast or fixed-route type, the point-to-point (P2P) handles point-to-point packets while the NN router handles nearest-neighbour packets and also deals with default and error routing. The output of the router stage is a vector of destinations to which the packet should be relayed. The third stage is the emergency routing mechanism for handling failed or congested links, which it detects using 'full' signals fed back from the individual destination output buffers.

10.4 Multicast (MC) router

The MC router uses the routing key in the MC packet to determine how to route the packet. The router has *1,024* look-up entries, each of which has a mask, a key value, and an output vector. The packet's routing key is compared with each entry in the MC router. For each entry it is first ANDed with the mask, then compared with the entry's key. If it matches, the entry's output vector is used to determine where the packet is sent; it can be sent to any subset (including all) of the local processors and the output links.

Thus, to programme an MC entry three writes are required: to the key, its mask and the corresponding vector. A mask of FFFFFFFF ensures all the key bits are used; if any mask bits are '0' the corresponding key bits should also be '0', otherwise the entry will not match. This can be

exploited to ensure that unused entries are invalid. The effect of the various combinations of bit values in the mask[] and key[] regions is summarized in the table below:

key[]	mask[]	Function
0	0	don't care - bit matches
1	0	bit misses - entry invalidated
0	1	match 0
1	1	match 1

Thus a particular entry [i] will match only if:

- wherever a bit in the mask[i] word is 1, the corresponding bit in the MC packet routing word is the same as the corresponding bit in the key[i] word, AND
- wherever a bit in the mask[i] word is 0, the corresponding bit in the key[i] word is also 0.

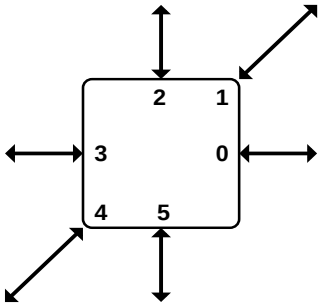
Note that the MC Router CAM is not initialised at reset. Before the Router is enabled all CAM entries must be initialised by software. Unused mask[] entries should be initialised to 0000000, and unused key[] entries should be initialised to FFFFFFFF. This invalidates every bit in the word, ensuring that the word will miss even in the presence of minor component failures.

The matching is perfomed in a parallel ternary associative memory, with a RAM used to store the output vectors. The associative memory can be set up so that more than one entry matches an incoming routing key; in this case the matching entry at the lowest address determines the output vector to be used. Multiple simultaneous matches can also be used to improve test efficiency.

If no entry matches an MC packet's routing key then default routing is employed - the packet is sent to the output link opposite the input link through which it arrived. Packets from local processors cannot be default-routed; the router table must have a valid entry for every locally-sourced packet.

The MC output vector assignment is detailed in the table below:

MC vector entry	Output port	Direction
bit[0]	Tx0	East
bit[1]	Tx1	North-East
bit[2]	Tx2	North
bit[3]	Tx3	West
bit[4]	Tx4	South-West
bit[5]	Tx5	South
bit[6]	Processor 0	Local
bit[7]	Processor 1	Local
...	...	...
bit[23]	Processor 17	Local



Link directions

If any of the multicast packet's output links are blocked the packet is stalled for a time 'wait1' (see 'r0: Router control register' on page 44). When that time expires any blocked external outputs (i.e. links 0-5) will attempt to divert to the next lower number link, modulo 6 (see section 10.9 on page 42) and retry for a further period, 'wait2'. If two potential outputs become unblocked at the

same time the original choice is preferred.

A packet which is diverted is typed as specified in ‘emergency routing’ on page 32. If a packet of such a type is received the router will attempt to output it as a ‘reverting’ packet to the output with the next lower number to the input on which it was received. If this should also be a normal packet then conventional multicast routing also takes place.

The routing tables should not be set up so that a packet paths cross each other. If the packet is programmed to do this then it is not possible to differentiate between an intended and a reverting packet; the ‘reverting’ designation takes priority.

A received reverting packet is routed normally if it is recognised by the router, otherwise it is ‘default’ routed to the link numbered two greater (mod 6) than the input link.

fixed-route (FR) packets

The FR router uses the same mechanism as the MC router although the packets do not have a key field. Instead, all packets of this type are routed to the same output vector, as specified in r33. Emergency routing is handled identically to MC packets.

This mechanism is intended to facilitate monitoring and debugging by routing data towards a point which connects with a host system.

10.5 The point-to-point (P2P) router

The P2P router uses the 16-bit destination ID in a point-to-point packet to determine which output the packet should be routed to. There is a 3-bit entry for each of the 64K destination IDs. Each 3-bit entry is decoded to determine whether the packet is delivered to the local Monitor Processor or one of the six output links, or dropped, as detailed in the table below:

P2P table entry	Output port	Direction
000	Tx0	East
001	Tx1	North-East
010	Tx2	North
011	Tx3	West
100	Tx4	South-West
101	Tx5	South
110	none (drop packet)	none
111	Monitor Processor	Local

The 3-bit entries are packed into an 8K entry x 24-bit SRAM lookup table. The 24-bit words hold entries 0, 8, 16, ... in bits [2:0], 1, 9, 17, ... in bits [5:3], etc.

10.6 The nearest-neighbour (NN) router

Nearest-neighbour packets are used to initialise the system and to perform run-time flood-fill and debug functions. The routing function here is to send ‘normal’ NN packets that arrive from outside the node (i.e. via an Rx link) to the monitor processor and to send NN packets that are generated internally to the appropriate output (Tx) link(s). This is to support a flood-fill load process.

In addition, the ‘peek/poke’ form of NN packet can be used by neighbouring systems to access System NoC resources. Here an NN poke ‘write’ packet (which is a peek/poke type with a 32-bit payload) is used to write the 32-bit data defined in the payload to a 32-bit address defined in the address/operation field. An NN peek ‘read’ packet (which is a peek/poke type without a 32-bit

payload) uses the 32-bit address defined in the address/operation field to read from the System NoC and returns the result (as a 'normal' NN packet) to the neighbour that issued the original packet using the Rx link ID to identify that source. This 'peek/poke' access to a neighbouring chip's principal resources can be used to investigate a non-functional chip, to re-assign the Monitor Processor from outside, and generally to get good visibility into a chip for test and debug purposes.

As the peek/poke NN packets convey only 32-bit data payloads the bottom 2 bits of the address should always be zero. All peek/poke NN packets return a response to the sender, with bit 0 of the address set to 1. Bit 1 will also be set to 1 if there was a bus error at the target. Peeks return a 32-bit data payload; pokes return without a payload.

default and error routing

In addition, the NN router performs default and error routing functions.

10.7 Time phase handling

The Router maintains a 2-bit time phase signal that is used to delete packets that are out-of date. The time phase logic operates as follows:

- locally-generated packets will have the current time phase inserted (where appropriate);
- a packet arriving from off-chip will have its time phase checked, and if it is two phases old it will be deleted (dropped, and copied to the Error registers).

10.8 Packet error handler

The packet error handler is a routing engine that simply flags the packet for dropping to the Error registers if it detects any of the following:

- a packet parity error;
- a packet that is two time phases old;
- a packet that is the wrong length.

The Monitor Processor can be interrupted to deal with packets dropped with errors.

10.9 Emergency routing

If a link fails (temporarily, due to congestion, or permanently, due to component failure) action will be taken at two levels:

- The blocked link will be detected in hardware and subsequent packets rerouted via the other two sides of a triangle of which the suspect link was an edge, being initially re-routed via the link which is rotated one link clockwise from the blocked link (so if link Tx0 fails, link Tx5 is used, etc).
- The Monitor Processor will be informed. It can track the problem using a diagnostic counter:
  - if the problem was due to transient congestion, it will note the congestion but do nothing further;
  - if the problem was due to recurring congestion, it will negotiate and establish a new route for some of the traffic using this link;
  - if the problem appears permanent, it will establish new routes for all of the traffic using this link.

  The hardware support for these processes include:
  - default routing processes in adjacent nodes that are invoked by flagging the packet as an emergency type;
  - mechanisms to inform the Monitor Processor of the problem;
  - means of inducing the various types of fault for testing purposes.

  Emergency rerouting around the triangle requires additional emergency packet types for MC and FR packets. P2P packets will find their own way to their destination following emergency routing.

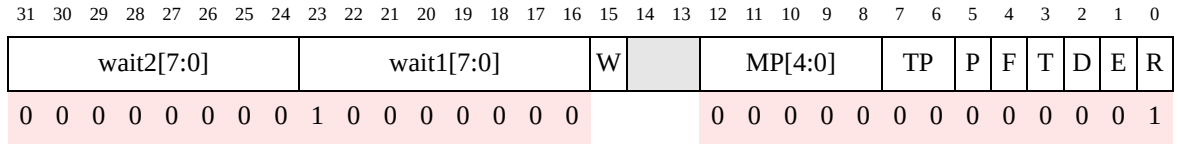
10.10 Register summary

Base address: 0xe1000000 (buffered write), 0xf1000000 (unbuffered write).

Name	Offset	R/W	Function
r0: control	0x00	R/W	Router control register
r1: status	0x04	R	Router status
r2: error header	0x08	R	error packet control byte and flags
r3: error routing	0x0C	R	error packet routing word
r4: error payload	0x10	R	error packet data payload
r5: error status	0x14	R	error packet status
r6: dump header	0x18	R	dumped packet control byte and flags
r7: dump routing	0x1C	R	dumped packet routing word
r8: dump payload	0x20	R	dumped packet data payload
r9: dump outputs	0x24	R	dumped packet intended destinations
r10: dump status	0x28	R	dumped packet status
r11: diag enables	0x2C	R/W	diagnostic counter enables
r12: timing ctr ctl	0x30	R/W	timing counter controls
r13: cycle ctr	0x34	R	counts Router clock cycles
r14: busy cyc ctr	0x38	R	counts emergency router active cycles
r15: no wt pkt ctr	0x3C	R	counts packets that do not wait to be issued
r16-31: dly hist	0x40-7C	R	packet delay histogram counters
r32: diversion	0x80	R/W	divert default packets
r33: FR route	0x84	R/W	fixed-route packet routing vector
rFN: diag filter	0x200-23C	R/W	diagnostic count filters (N = 0-15)
rCN: diag count	0x300-33C	R/W	diagnostic counters (N = 0-15)
rT1: test register	0xF00	R	hardware test register 1
rT2: test key	0xF04	R/W	hardware test register 2 - CAM input test key
route[1023:0]	0x4000	R/W	MC Router routing word values
key[1023:0]	0x8000	W	MC Router key values
mask[1023:0]	0xC000	W	MC Router mask values
P2P[8191:0]	0x10000	R/W	P2P Router routing entries (8 3-bit entries/word)

10.11 Register details

r0: Router control register



The functions of these fields are described in the table below:

Name	bits	R/W	Function
wait2[7:0]	31:24	R/W	wait time before dropping packet
wait1[7:0]	23:16	R/W	wait time before emergency routing
W	15	W	re-initialise wait counters
MP[4:0]	12:8	R/W	Monitor Processor ID number
TP	7:6	R/W	time phase (c.f. packet time stamps)
P	5	R/W	enable count of packet parity errors
F	4	R/W	enable count of packet framing errors
T	3	R/W	enable count of packet time stamp errors
D	2	R/W	enable dump packet interrupt
E	1	R/W	enable error packet interrupt
R	0	R/W	enable packet routing

The wait times (defined by wait1[] and wait2[]) are stored in a floating point format to give a wide range of values with high accuracy at low values combined with simple implementation using a binary pre-scaler and a loadable counter. Each 8-bit field is divided into a 4-bit mantissa M[3:0] = wait[3:0] and a 4-bit exponent E[3:0] = wait[7:4]. The wait time in clock cycles is then given by:

$$wait = (M + 16 - 2^{4-E}).2^{E} \quad \text{for } E \leq 4;$$

$$wait = (M + 16).2^{E} \quad \text{for } E > 4;$$

Note that wait[7:0] = 0x00 gives a wait time of zero, and the wait time increases monotonically with wait[7:0]; wait[7:0] = 0xFF is a special case and gives an infinite wait time - wait forever.

There is a small semantic difference between wait1[7:0] and wait2[7:0]:

- wait1[7:0] defines the number of cycles the Router will re-try after the first failed cycle before attempting emergency routing; wait1[] = 0 will attempt normal routing once and then try emergency routing.
- wait2[7:0] is the number of cycles during which emergency routing will be attempted before the packet is dumped; wait2[] = 0 therefore effectively disables emergency routing.

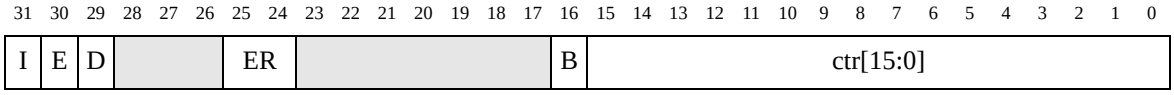
If r0 is written when one of the wait counters is running, writing a 1 to W (bit[15]) will cause the active counter to restart from the new value written to it. This enables the Monitor Processor to clear a deadlocked ‘wait forever’ condition. If 0 is written to W the active counter will not restart

but will use the new wait time value the next time it is invoked.

Note that the Router is enabled after reset. This is so that a neighbouring chip can peek and poke a chip that fails after reset using NN packets, to diagnose and possibly fix the cause of failure.

r1: Router status

All Router interrupt request sources are visible here, as is the current status of the emergency routing system.



The functions of these fields are described in the table below:

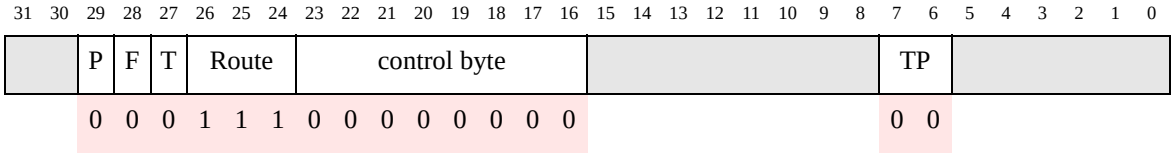
Name	bits	R/W	Function
I: interrupt active	31	R	combined Router interrupt request
E: error int	30	R	error packet interrupt active
D: dump int	29	R	dump packet interrupt active
ER[1:0]	25:24	R	Router output stage status (empty, full but unblocked, blocked in wait1, blocked in wait2)
B	16	R	busy - active packet(s) in Router pipeline
ctr[15:0]	15:0	R	diagnostic counter interrupt active

The Router generates three interrupt request outputs that are handled by the VIC on each processor: diagnostic counter event interrupt, dump interrupt and error interrupt. These correspond to the OR of ctr[15:0], D and E respectively.

The interrupt requests are cleared by reading their respective status registers: r5, r10 and r2N.

r2: error header

A packet which contains an error is copied to r2-5. Once a packet has been copied (indicated by bit[31] of r5 being set) any further error packet is ignored, except that it can update the sticky bits in r5 (and errors of the types specified in r0 are counted in r5).



The functions of these fields are described in the table below:

Name	bits	R/W	Function
P: parity	29	R	packet parity error
F: framing error	28	R	packet framing error
T: TP error	27	R	packet time stamp error

Name	bits	R/W	Function
Route	26:24	R	Rx route field of error packet
Control byte	23:16	R	control byte of error packet
TP: time phase	7:6	R	time phase when packet received

r3: error routing word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit routing word																															

r4: error data payload

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit data payload																															

r5: error status

This register counts error packets, including time stamp, framing and parity errors as enabled by r0[5:3]. The Monitor Processor resets r5[31:27] and the error count by reading its contents.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	V	P	F	T												error count															
0	0	0	0	0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
E: error	31	R	error packet detected
V: overflow	30	R	more than one error packet detected
P: parity	29	R	packet parity error (sticky)
F: framing error	28	R	packet framing error (sticky)
T: TP error	27	R	packet time stamp error (sticky)
error count	15:0	R	16-bit saturating error count

r6: dump header

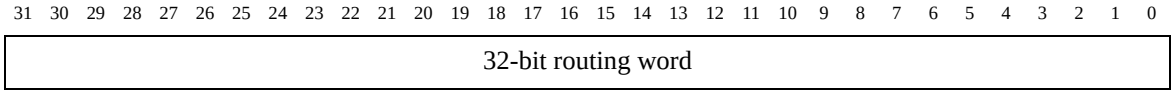
A packet which is dumped because it cannot be routed to its destinations is copied to r6-10. Once a packet has been dumped (indicated by bit[31] of r10 being set) any further packet that is dumped is ignored, except that it can update the sticky bits in r10 (and can be counted by a diagnostic counter).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Route			control byte																TP							
reset:					1	1	1	0	0	0	0	0	0	0	0									0	0						

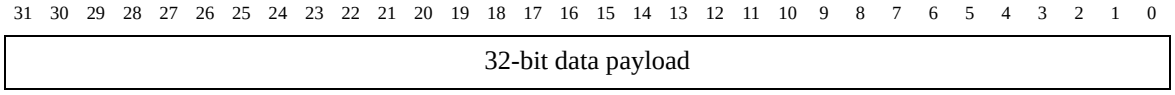
The functions of these fields are described in the table below:

Name	bits	R/W	Function
Route	26:24	R	Rx route field of dumped packet
Control byte	23:16	R	control byte of dumped packet
TP: time phase	7:6	R	time phase when packet dumped

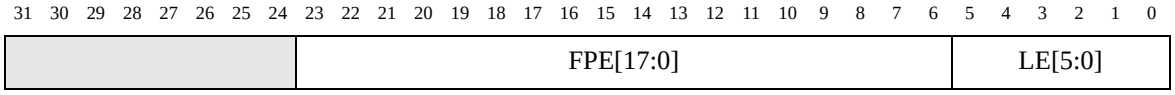
r7: dump routing word



r8: dump data payload



r9: dump outputs

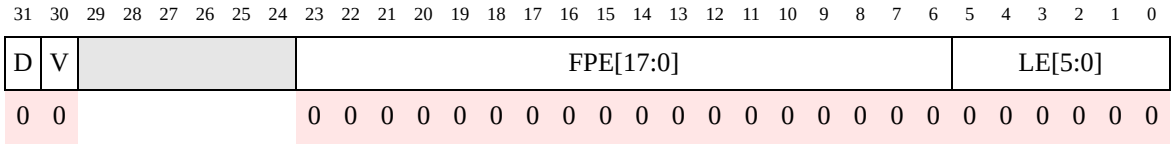


The functions of these fields are described in the table below:

Name	bits	R/W	Function
FPE[17:0]	23:6	R	Fascicle Processor link error caused dump
LE[5:0]	5:0	R	Tx link transmit error caused packet dump

r10: dump status

The Monitor Processor resets r10 by reading its contents.



The functions of these fields are described in the table below:

Name	bits	R/W	Function
D: dumped	31	R	packet dumped
V: overflow	30	R	more than one packet dumped
FPE[17:0]	23:6	R	Fascicle Proc link error caused dump (sticky)
LE[5:0]	5:0	R	Tx link error caused dump (sticky)

r11: diagnostic counter enable/reset

This register provides a single control point for the 16 diagnostic counters, enabling them to count events over a precisely controlled time period.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reset[15:0]																enable[15:0]															
reset:																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
reset[15:0]	31:16	W	write a 1 to reset diagnostic counter 15..0
enable[15:0]	15:0	R/W	enable diagnostic counter15..0

Writing a 0 to reset[15:0] has no effect. Writing a 1 clears the respective counter.

r12: timing counter controls

This register controls the cycle counters in registers r13, r14 & r15, and in the delay histogram registers r16-r31.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													T	S	R														H	E	C
reset:																													0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
T	18	W	reset histogram
S	17	W	reset emergency router active cycle counter
R	16	W	reset cycle counter
H	2	R/W	enable histogram
E	1	R/W	enable emergency router active cycle counter
C	0	R/W	enable cycle counter

Writing a 0 to R, S or T has no effect. Writing a 1 clears the respective counter.

r13: cycle count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit non-saturating cycle counter																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

r13, when enabled by r12, simply counts the number of Router clock cycles.

r14: emergency router active cycle count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit non-saturating emergency router active cycle counter																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

r14, when enabled by r12, counts the number of cycles for which the emergency router is actively seeking a route for a packet. This equals the number of packets plus the number of stall cycles.

r15: unblocked packet count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit non-saturating unblocked packet counter																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

r15, when enabled by r12, counts the number of packets which pass through undelayed by congested output links.

r16-31: packet delay histogram

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit non-saturating packet delay counter																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

r16-r31, when enabled by r12, count the number of times a packet is delayed due to link congestion, each register counting delays within a range of clock cycles. r15 counts the zero delay component of the histogram. These counters use the same pre-scaling as wait1 in r0, so the histogram effectively records the value in the wait mantissa at the time the congestion resolves.

r32: diversion

This register allows default-routed MC packets to be redirected in the case when their default path is unavailable, for example as a result of a complete node failure.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																				L5	L4		L3		L2		L1		L0		
reset:																				0	0	0	0	0	0	0	0	0	0	0	0

The 2-bit L0 field can be set to 00 for normal behaviour of packets default routed from link 0, to x1 to divert those packets to the local Monitor Processor, or to 10 to destroy the packets. L1 likewise controls default routed packets that arrive through link 1, etc.

r33: fixed-route packet routing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NN broadcast								FR output vector																							
1	1	1	1	1	1			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

r33 routes fixed-route (type 3) packets to off-chip links and local processors in exactly the same

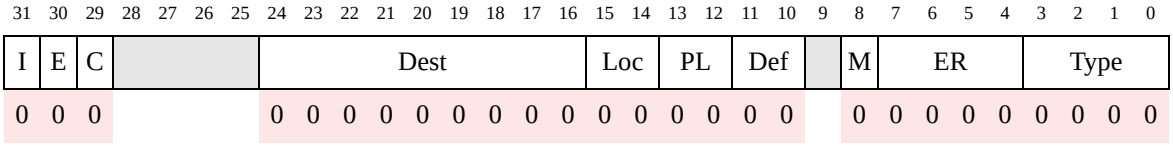
way, with the same bit allocation, as an MC output vector as described in section 10.4 on page 39.

In addition, the 'NN broadcast' bits[31:26] define which links an NN broadcast packet is sent through. A 1 indicates an active link, and bit[26] is for link 0, bit[27] link 1, etc.

rFN: diagnostic filter control

The Router has 16 diagnostic counters (N = 0..F) each of which counts packets passing through the Router filtered on packet characteristics defined here. A packet is counted if it has characteristics that match with a '1' in each of the 6 fields. Setting all bits [24:10, 7:0] to '1' will count all packets.

A diagnostic counter may (optionally) generate an interrupt on each count. The C bit[29] is a sticky bit set when a counter event occurs and is cleared whenever this register is read.



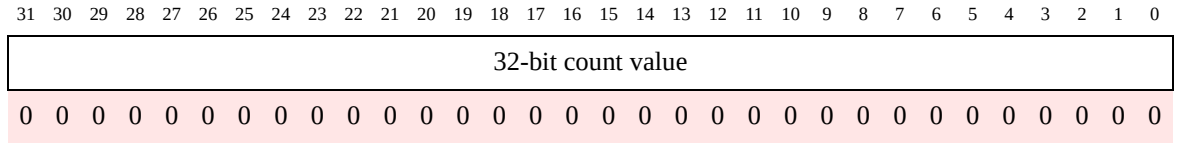
The functions of these fields are described in the table below:

Name	bits	R/W	Function
I	31	R	counter interrupt active: I = E AND C
E	30	R/W	enable interrupt on counter event
C	29	R	counter event has occurred (sticky)
Dest	24:16	R/W	packet dest (Tx link[5:0], MP, local ¬MP, dump)
Loc	15:14	R/W	local [x1]/non-local[1x] packet source
PL	13:12	R/W	packets with [x1]/without [1x] payload
Def	11:10	R/W	default [x1]/non-default [1x] routed packets
M	8	R/W	Emergency Routing mode
ER	7:4	R/W	Emergency Routing field = 3, 2, 1 or 0
Type	3:0	R/W	packet type: fr, nn, p2p, mc

If M (bit[8]) = 0 the Emergency Routing field matches that of the incoming packet, before any local Emergency Routing, so this can be used to count packets that have been Emergency Routed by a previous Router but not those that are Emergency Routed here.

If M = 1 the Emergency Routing field is matched against outgoing packets to destinations selected in the Dest field. If any outgoing packet to a selected destination matches the ER field the diagnostic count will be incremented. (Note that packets to internal destinations cannot be emergency routed and so have ER = 0.)

rCN: diagnostic counters

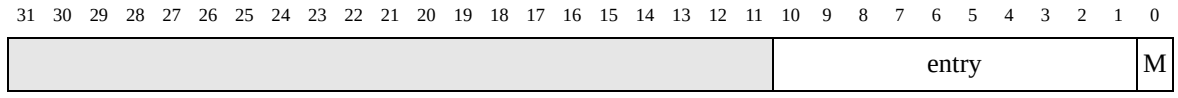


Each of these counters can be used to count selected types of packets under the control of the corresponding rFN. The counter can have any value written to it, and will increment from that value when respective events occur.

If an event occurs as the counter is being written it will not be counted. To avoid missing an event it is better to avoid writing the counter; instead, read it at the start of a time period and subtract this value from the value read at the end of the period to get a count of the number of events during the period.

rT1: hardware test register 1

This register is used only for hardware test purposes, and has no useful functions for the application programmer.

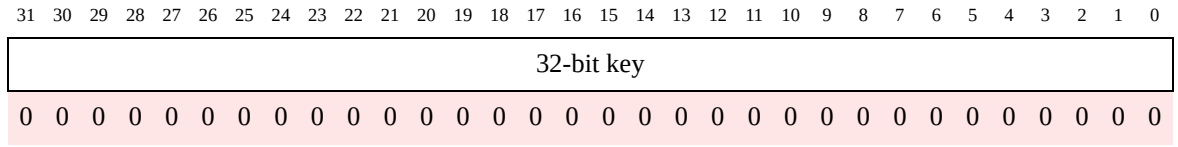


The functions of these fields are described in the table below:

Name	bits	R/W	Function
M	0	R	MC router associative look-up ‘miss’ output
entry	10:1	R	MC router associative look-up entry address

The input key used for the associative look-up whenever this register is read is in register T2.

rT2: hardware test register 2



This register holds the key presented to the association input of the multicast router when register T1 is read.

10.12 Fault-tolerance

The Communications Router has some internal fault-tolerance capacity, in particular it is possible to map out a failed multicast router entry. This is a useful mechanism as the multicast router dominates the silicon area of the Communications Router.

There is also capacity to cope with external failures. Emergency routing will attempt to bypass a faulty or blocked link. In the event of a node (or larger) failure this will not be sufficient. In order to tolerate a chip failure several expedients can be employed on a local basis:

- P2P packets can be routed around the obstruction;
- MC packets with a router entry can be redirected appropriately.

In most cases, default MC packets cannot sensibly be trapped by adding table entries due to their (almost) infinite variety. To allow rerouting, these packets can be dropped to the Monitor Processor on a link-by-link basis using the diversion register. In principle they can then be routed around the obstruction as P2P payloads before being resurrected at the opposite side.

Should the Monitor Processor become overwhelmed, it is also possible to use the diversion register to eliminate these packets in the Router; this prevents them blocking the Router pipeline whilst waiting for a timeout and thus delaying viable traffic.

Fault detection

- packet parity errors.
- packet time-phase errors.
- packet unroutable errors (e.g. a locally-sourced multicast packet which doesn't match any entry in the multicast router).
- wrong packet length.

Fault isolation

- a multicast router entry can be disabled if it fails - see initialisation guidance above.

Reconfiguration

- since all multicast router entries are identical the function of any entry can be relocated to a spare entry.
- if a router becomes full a global reallocation of resources can move functionality to a different router.

10.13 Test

Production test

The ternary CAM used in the multicast router has access for parallel testing, so a processor can write a value to all locations and see if an input with 1 bit flipped results in a hit or a miss. The CAM is not directly readable - attempts to read this space will result in bus errors - and must be tested by association. To do this a key must first be written into register rT2. A subsequent read of register rT1 will then indicate if that key has associated with any CAM entries. If it has not then rT1<0> will be set and the other bits of this register will be undefined; if one or more of the entries are matched then the one at the lowest address in the CAM will be indicated in the 'entry' field.

All RAMs have read-write access for test purposes.

11. Inter-chip transmit and receive interfaces

Inter-chip communication is implemented by extending the Communications NoC from chip to chip. In order to sustain throughput, there is a protocol conversion at each chip boundary from standard CHAIN 3-of-6 return-to-zero to 2-of-7 non-return-to-zero. The interfaces include logic to minimise the risk of a protocol deadlock caused by glitches on the inter-chip wires.

11.1 Features

- transmit (Tx) interface:
  - converts on-chip 3-of-6 RTZ symbol into off-chip 2-of-7 NRZ symbol;
  - disable control input;
  - reset input.
- receive (Rx) interface:
  - converts off-chip 2-of-7 NRZ symbol into on-chip 3-of-6 RTZ symbol;
  - disable control input;
  - reset input.

11.2 Programmer view

The only programmer-accessible features implemented in these interfaces are software reset and a disable control, both accessed via the System Controller. In normal operation these interfaces provide transparent connectivity between the routing network on one chip and those on its neighbours.

11.3 Fault-tolerance

The fault inducing, detecting and resetting functions are controlled from the System Controller (see ‘System Controller’ on page 66). The interfaces are ‘glitch hardened’ to greatly reduce the probability of a link deadlock arising as a result of a glitch on one of the inter-chip wires. Such a glitch may introduce packet errors, which will be detected and handled elsewhere, but it is very unlikely to cause deadlock. It is expected that the link reset function will not be required often.

Fault insertion

- an input controlled by the System Controller causes the interface to deadlock (by disabling it).

Fault detection

- Monitor Processors should regularly test link functionality.

Fault isolation

- the interface can be disabled to isolate the chip-to-chip link. This input from the System Controller is also used to create a fault.

Reconfiguration

- the link interface can be reset by the System Controller to attempt recovery from a fault.
- the link interface can be isolated and an alternative route used.

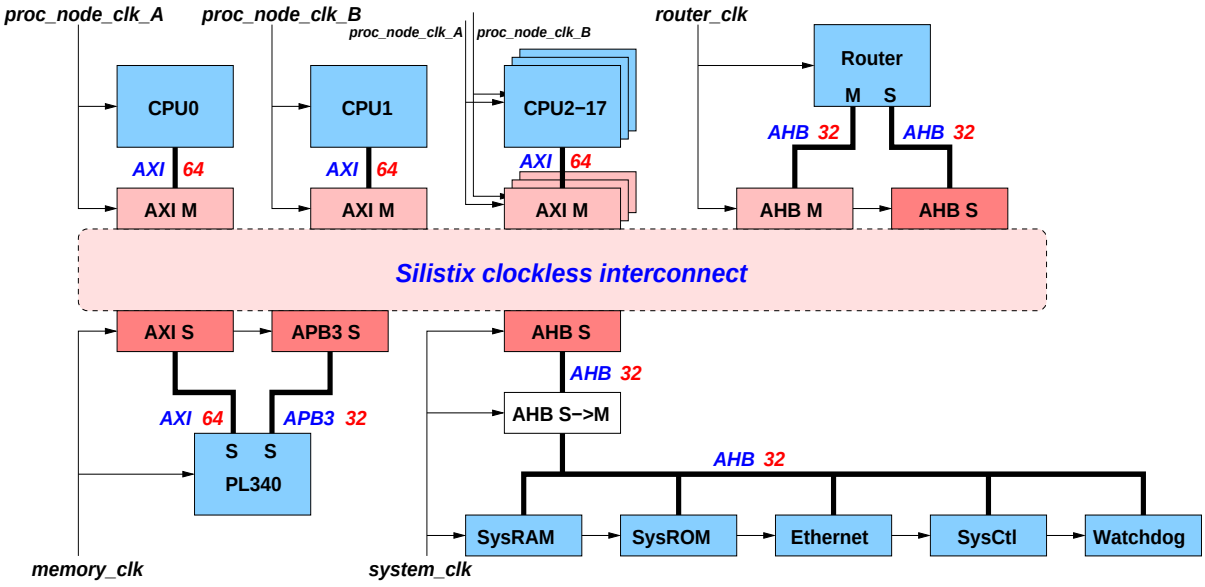
12. System NoC

The System NoC has a primary function of connecting the ARM968 processors to the SDRAM interface. It is also used to connect the processors to system control and test functions, and for a variety of other purposes.

12.1 Features

- supports full bandwidth block transfers between the SDRAM and the ARM968 processors.
- the Router is an additional initiator for system debug purposes.
- can be reset (in subsections) to clear deadlocks.
- multiple targets:
  - SDRAM interface - ARM PL340
  - System RAM
  - System ROM
  - Ethernet interface
  - System Controller
  - Watchdog Timer.
  - Router configuration registers

12.2 Organisation



13. SDRAM interface

The SDRAM interface connects the System NoC to an off-chip SDRAM device. It is the ARM PL340, described in ARM document DDI 0331D.

13.1 Features

- control for external Mobile DDR SDRAM memory device
- memory request queue
- out of order request sequencing to maximise memory throughput
- AXI interface to System NoC
- delay-locked loop (DLL) to realign SDRAM data strobes with the input data streams

13.2 Register summary

Base address: 0xe0000000 (buffered write), 0xf0000000 (unbuffered write).

User registers

The following registers allow normal user programming of the PL340 SDRAM interface:

Name	Offset	R/W	Function
r0: status	0x00	R	memory controller status
r1: command	0x04	W	PL340 command
r2: direct	0x08	W	direct command
r3: mem_cfg	0x0C	R/W	memory configuration
r4: refresh_prd	0x10	R/W	refresh period
r5: CAS_latency	0x14	R/W	CAS latency
r6: t_dqss	0x18	R/W	write to DQS time
r7: t_mrd	0x1C	R/W	mode register command time
r8: t_ras	0x20	R/W	RAS to precharge delay
r9: t_rc	0x24	R/W	active bank x to active bank x delay
r10: t_rcd	0x28	R/W	RAS to CAS minimum delay
r11: t_rfc	0x2C	R/W	auto-refresh command time
r12: t_rp	0x30	R/W	precharge to RAS delay
r13: t_rrd	0x34	R/W	active bank x to active bank y delay
r14: t_wr	0x38	R/W	write to precharge delay
r15: t_wtr	0x3C	R/W	write to read delay
r16: t_xp	0x40	R/W	exit power-down command time
r17: t_xsr	0x44	R/W	exit self-refresh command time

Name	Offset	R/W	Function
r18: t_esr	0x48	R/W	self-refresh command time
id_n_cfg	0x100	R/W	QoS settings
chip_n_cfg	0x200	R/W	external memory device configuration
user_status	0x300	R	DLL test and status inputs
user_config0	0x304	W	DLL test and control outputs
user_config1	0x308	W	DLL fine-tune control

Test and ID registers

In addition, there are test and ID registers that will not normally be of interest to the programmer:

Name	Offset	R/W	Function
int_cfg	0xE00	R/W	integration configuration register
int_inputs	0xE04	R	integration inputs register
int_outputs	0xE08	W	integration outputs register
periph_id_n	0xFE0-C	R	PL340 peripheral ID byte registers
pcell_id_n	0xFF0-C	R	PL340 Prime Cell ID byte registers

See ARM document DDI 0331D for further details of the test registers.

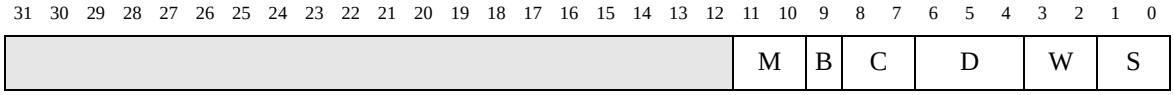
Restrictions on when registers may be modified

Normally the PL340 registers will be initialised during system start-up and then left alone. Restrictions on when the registers may be safely modified are detailed in the PL340 datasheet, ARM doccument DDI 0331D.

The DLL test and control outputs and the DLL fine-tune control registers should only be written to when the PL340 is quiescent and no processor is issuing an SDRAM access or has one pending.

13.3 Register details

r0: memory controller status

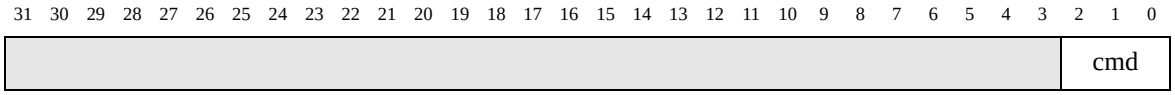


The functions of these fields are described in the table below:

Name	bits	R/W	Function
M: monitors	11:10	R	Number of exclusive access monitors (0, 1, 2, 4)
B: banks	9	R	Fixed at 1’b01 = 4 banks on a chip
C: chips	8:7	R	Number of different chip selects (1, 2, 3, 4)

Name	bits	R/W	Function
D: DDR	6:4	R	DDR type: 3b'011 = Mobile DDR
W: width	3:2	R	Width of external memory: 2'b01 = 32 bits
S: status	1:0	R	Config, ready, paused, low-power

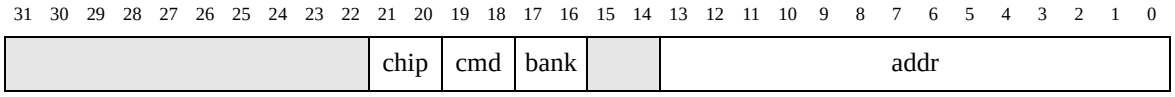
r1: memory controller command



The function of this field is described in the table below:

Name	bits	R/W	Function
cmd: command	2:0	W	Go, sleep, wake-up, pause, config, active_pause

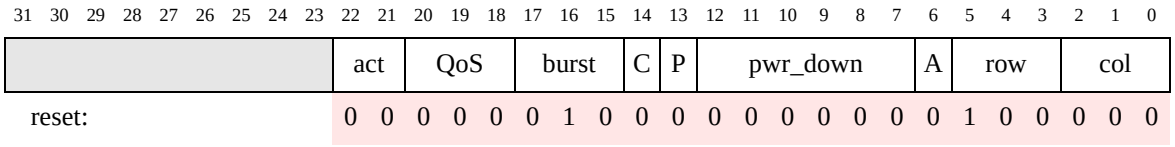
r2: direct command



This register is used to pass a command directly to a memory device attached to the PL340. The functions of these fields are described in the table below:

Name	bits	R/W	Function
chip	21:20	W	chip number
cmd	19:18	W	command passed to memory device
bank	17:16	W	bank passed to memory device
addr[13:0]	13:0	W	address passed to memory device

r3: memory configuration



This register is used to describe the configuration of the memory device(s) attached to the PL340. The functions of these fields are described in the table below:

Name	bits	R/W	Function
act	22:21	R/W	active chips: number for refresh generation
QoS	20:18	R/W	selects the 4-bit QoS field from the AXI ARID

Name	bits	R/W	Function
burst	17:15	R/W	burst length (1, 2, 4, 8, 16)
C	14	R/W	stop memory clock when no access
P	13	R/W	auto-power-down memory when inactive
pwr_down	12:7	R/W	# memory cycles before auto-power-down
A	6	R/W	position of auto-pre-charge bit (10/8)
row	5:3	R/W	number of row address bits (11-16)
col	2:0	R/W	number of column address bits (8-12)

r4: refresh period

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	refresh period														
reset:																	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0

The function of this field is described in the table below:

Name	bits	R/W	Function
refresh period	14:0	R/W	memory refresh period in memory clock cycles

r5: CAS latency

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												cas_lat			H
reset:																												0	1	1	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
cas_lat	3:1	R/W	CAS latency in memory clock cycles
H	0	R/W	CAS half cycle - must be set to 1’b0

r6: t_dqss

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															tdqss
reset:																														0	1

The function of this field is described in the table below:

Name	bits	R/W	Function
tdqss	1:0	R/W	write to DQS in memory clock cycles

r7: t_mrd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									t_mrd						
reset:																									0	0	0	0	0	1	0

The function of this field is described in the table below:

Name	bits	R/W	Function
t_mrd	6:0	R/W	mode reg cmnd time in memory clock cycles

r8: t_ras

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												t_ras			
reset:																												0	1	1	1

The function of this field is described in the table below:

Name	bits	R/W	Function
t_ras	3:0	R/W	RAS to precharge time in memory clock cycles

r9: t_rc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												t_rc			
reset:																												1	0	1	1

The function of this field is described in the table below:

Name	bits	R/W	Function
t_rc	3:0	R/W	Bank x to bank x delay in memory clock cycles

r10: t_rcd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										sched			t_rcd		
reset:																										0	1	1	1	0	1

The functions of these fields are described in the table below:

Name	bits	R/W	Function
t_rcd	2:0	R/W	RAS to CAS min delay in memory clock cycles
sched	5:3	R/W	RAS to CAS min delay in **aclk** cycles -3

r11: t_rfc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										sched				t_rfc			
reset:																						1	0	0	0	0	1	0	0	1	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
sched	9:5	R/W	Auto-refresh cmnd time in **aclk** cycles -3
t_rfc	4:0	R/W	Auto-refresh cmnd time in memory clock cycles

r12: t_rp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										sched			t_rp		
reset:																										0	1	1	1	0	1

The functions of these fields are described in the table below:

Name	bits	R/W	Function
sched	5:3	R/W	Precharge to RAS delay in **aclk** cycles -3
t_rp	2:0	R/W	Precharge to RAS delay in memory clock cycles

r13: t_rrd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												t_rrd			
reset:																												0	0	1	0

The function of this field is described in the table below:

Name	bits	R/W	Function
t_rrd	3:0	R/W	Bank x to bank y delay in memory clock cycles

r14: t_wr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													t_wr		
reset:																													0	1	1

The function of this field is described in the table below:

Name	bits	R/W	Function
t_wr	2:0	R/W	Write to precharge dly in memory clock cycles

r15: t_wtr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													t_wtr		
reset:																													0	1	0

The function of this field is described in the table below:

Name	bits	R/W	Function
t_wtr	2:0	R/W	Write to read delay in memory clock cycles

r16: t_xp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														t_xp	
reset:																								0	0	0	0	0	0	0	1

The function of this field is described in the table below:

Name	bits	R/W	Function
t_xp	7:0	R/W	Exit pwr-dn cmnd time in memory clock cycles

r17: t_xsr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								t_xsr							
reset:																								0	0	0	0	1	0	1	0

The function of this field is described in the table below:

Name	bits	R/W	Function
t_xsr	7:0	R/W	Exit self-rfsh cmnd time in mem clock cycles

r18: t_esr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								t_esr							
reset:																								0	0	0	1	0	1	0	0

The function of this field is described in the table below:

Name	bits	R/W	Function
t_esr	7:0	R/W	Self-refresh cmnd time in memory clock cycles

id_n_cfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						QoS_max								N	E
reset:																						0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
QoS_max	9:2	R/W	maximum QoS
N	1	R/W	minimum QoS
E	0	R/W	QoS enable

chip_n_cfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															B	match								mask							
reset:															0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

There is one of these registers for each external chip that is supported. The functions of these fields

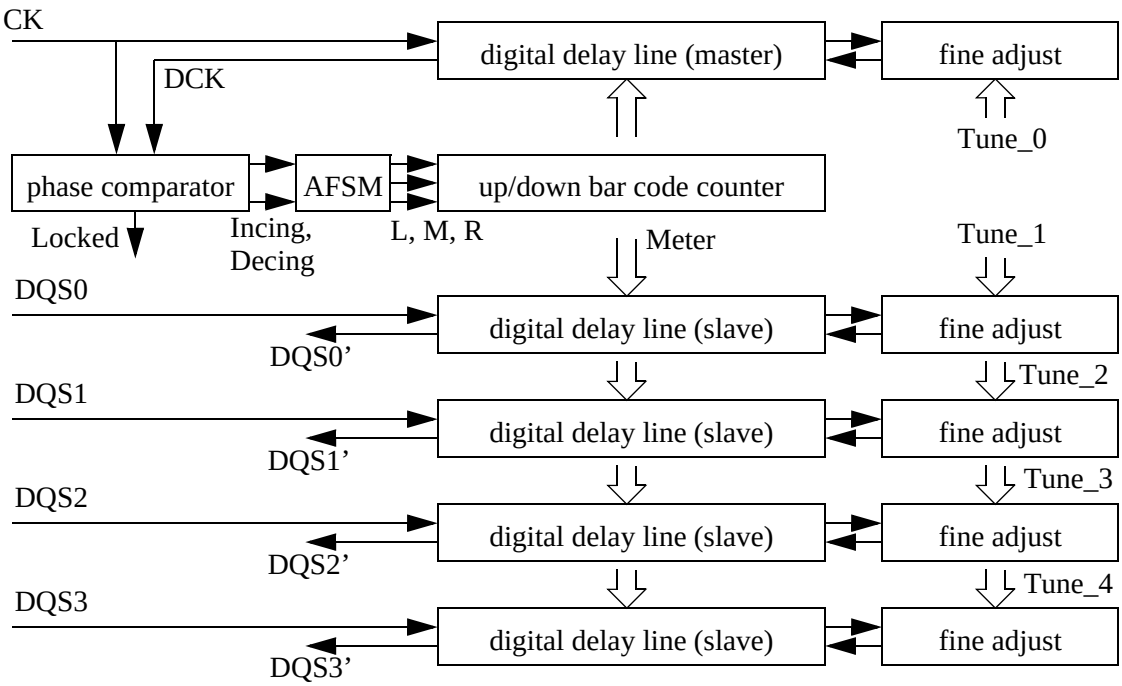
are described in the table below:

Name	bits	R/W	Function
B	16	R/W	bank-rol-column/row-bank-column
match	15:8	R/W	address match
mask	7:0	R/W	address mask

13.4 The delay-locked loop (DLL)

The SDRAM interface incorporates a delay-locked loop which, though outside the PL340, is controlled via the PL340 user status and configuration registers.

The general organisation of the DLL is shown below:



The basic operation is that a reference clock, CK, running at twice the SDRAM clock (i.e. nominally 333 MHz for a 166 MHz SDRAM), is passed through a master delay line and the output, DCK, inverted and compared with the original clock. A phase comparator drives an asynchronous finite state machine (AFSM) that in turn drives an up/down bar code counter to line these two signals up. The SDRAM data strobes, DQS0-3, are passed through matched delay lines to line up with the middle of the data valid period. Software can fine-tune the individual strobe timings.

There is a 6th, spare, delay line, that can be used if any of the five primary delay lines fails.

user_status: DLL test and status inputs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									L	M	R		K	I	D	C3	S3	C2	S2	C1	S1	C0	S0		Meter							
reset:									0	0	0		0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

The function of these fields is described in the table below:

Name	bits	R/W	Function
L, M, R	22:20	R	3-phase bar-code control output
K: locKed	18	R	Phase comparator is locked
I: Incing	17	R	Phase comparator is increasing delay
D: Decing	16	R	Phase comparator is reducing delay
C0, C1, C2, C3	9,11,13,15	R	Clock faster than strobe 0-3
S0, S1, S2, S3	8,10,12,14	R	Strobe 0-3 faster than Clock
Meter	6:0	R	Current position of bar-code output

user_config0: DLL test and control outputs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							E	TL	L	M	R	T5	ID	I	D					S5		S4		S3		S2		S1		S0		
reset:							0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0

The function of these fields is described in the table below:

Name	bits	R/W	Function
E: Enable	24	W	Enable DLL (0 = reset DLL)
TL: Test_LMR	23	W	Enable forcing of L, M, R
L, M, R	22:20	W	Force 3-phase bar-code control inputs
T5: Test_5	19	W	Substitute delay line 5 for 4 for testing
ID: Test_ID	18	W	Enable forcing of Incing and Decing
I: Test_Incing	17	W	Force Incing (if ID = 1)
D: Test_Decing	16	W	Force Decing (if ID = 1)
S0-S5	11:0	W	Input selects for the 6 delay lines{def, alt, 0, 1}

user_config1: DLL fine-tune control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Tune_5				Tune_4				Tune_3				Tune_2				Tune_1				Tune_0			
reset:								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The function of these fields is described in the table below:

Name	bits	R/W	Function
Tune0..5	23:0	W	Fine tuning control on delay lines 0..5

13.5 Fault-tolerance

Fault insertion

The DLL can be driven by software into pretty much any defective state.

Fault detection

The DLL delay lines can be tested for stuck-at faults and relative timing accuracy.

Fault isolation

A defective or out-or-spec delay line can be isolated.

Reconfiguration

A defective or out-or-spec delay line can be isolated and replaced by using the spare delay line.

14. System Controller

The System Controller incorporates a number of functions for system start-up, fault-tolerance testing (invoking, detecting and resetting faults), general performance monitoring, etc.

14.1 Features

- 'Arbiter' read-sensitive register bit to determine Monitor Processor ID at start-up.
- 32 test-and-set registers for general software use, e.g. to enforce mutually exclusive access to critical data structures.
- individual interrupt, reset and clock-enable controls and 'processor OK' status bits for each processor.
- sundry parallel IO and test and control registers.
- PLL and clock management registers.

14.2 Register summary

Base address: 0xe2000000 (buffered write), 0xf2000000 (unbuffered write).

These registers may only be accessed by a processor executing in a privileged mode; any attempt to access the System Controller from user-mode code will return a bus error. Only aligned word accesses are supported - misaligned word or byte or half-word accesses will return a bus error.

Name	Offset	R/W	Function
r0: Chip ID	0x00	R	Chip ID register (hardwired)
r1: CPU disable	0x04	R/W	Each bit disables the clock of a processor
r2: Set CPU IRQ	0x08	R/W	Writing a 1 sets a processor's interrupt line
r3: Clr CPU IRQ	0x0C	R/W	Writing a 1 clears a processor's interrupt line
r4: Set CPU OK	0x10	R/W	Writing a 1 sets a CPU OK bit
r5: Clr CPU OK	0x14	R/W	Writing a 1 clears a CPU OK bit
r6: CPU Rst Lv	0x18	R/W	Level control of CPU resets
r7: Node Rst Lv	0x1C	R/W	Level control of CPU node resets
r8: Sbsys Rst Lv	0x20	R/W	Level control of subsystem resets
r9: CPU Rst Pu	0x24	R/W	Pulse control of CPU resets
r10: Node Rst Pu	0x28	R/W	Pulse control of CPU node resets
r11: Sbsys Rst Pu	0x2C	R/W	Pulse control of subsystem resets
r12: Reset Code	0x30	R	Indicates cause of last chip reset
r13: Monitor ID	0x34	R/W	ID of Monitor Processor
r14: Misc control	0x38	R/W	Miscellaneous control bits
r15: GPIO pull u/d	0x3C	R/W	General-purpose IO pull up/down enable

Name	Offset	R/W	Function
r16: I/O port	0x40	R/W	I/O pin output register
r17: I/O direction	0x44	R/W	External I/O pin is input (1) or output (0)
r18: Set IO	0x48	R/W	Writing a 1 sets IO register bit
r19: Clear IO	0x4C	R/W	Writing a 1 clears IO register bit
r20: PLL1	0x50	R/W	PLL1 frequency control
r21: PLL2	0x54	R/W	PLL2 frequency control
r22: Set flags	0x58	R/W	Set flags register
r23: Reset flags	0x5C	R/W	Reset flags register
r24: Clk Mux Ctl	0x60	R/W	Clock multiplexer controls
r25: CPU sleep	0x64	R	CPU sleep (awaiting interrupt) status
r26-28	0x68-70	R/W	Temperature sensor registers [2:0]
r32-63: Arbiter	0x80-FC	R	Read sensitive semaphores to determine MP
r64-95: Test&Set	0x100-17C	R	Test & Set registers for general software use
r96-127: Test&Clr	0x180-1FC	R	Test & Clear registers for general software use
r128: Link disable	0x200	R/W	Disables for Tx and Rx link interfaces

14.3 Register details

r0: Chip ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
device												version				Year								#CPUs							
0	1	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0

This register is configured at chip design time to hold a unique ID for the chip type. The device code is 591 in BCD. The version will increment with each design variant. Year holds the last two digits of the year of first fabrication, in BCD. The bottom byte holds the number of CPUs on the chip.

The test chip ID is 0x59100902. The full chip ID is 0x59111012.

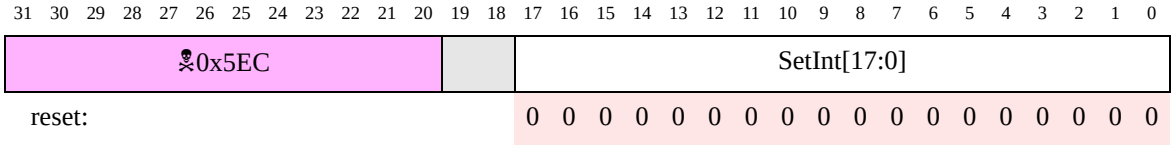
r1: CPU clock disable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
☠0x5EC														ClkDis[17:0]																	
reset:														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writing a 1 to bit[n] (n = 0..17) will disable the clock input to processor[n], thereby halting the processor indefinitely. Writing a 0 will enable the clock. For a write to be effective it must include a

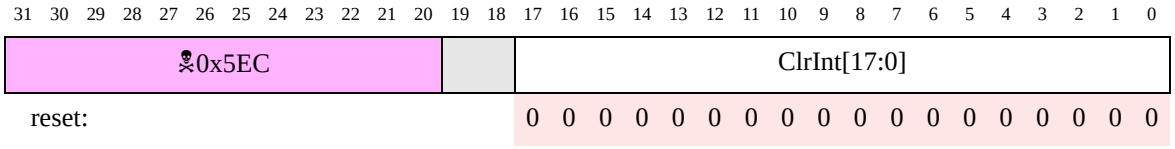
security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of all of the clock disable lines.

r2: Set CPU interrupt request



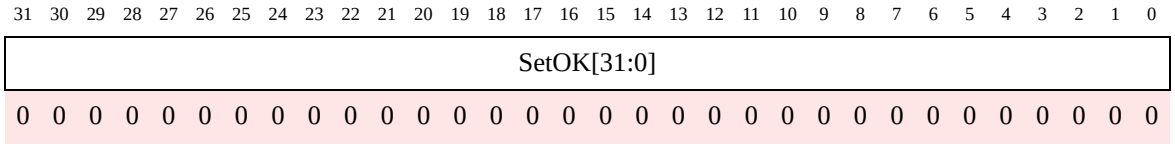
Writing a 1 to bit[n] (n = 0..17) will set an interrupt request to processor[n], which can be enabled/ disabled and routed to IRQ or FIQ by that processor's local Vectored Interrupt Controller (VIC - see page 12). Writing a 0 has no effect. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of all of the processor interrupt lines.

r3: Clear CPU interrupt request



Writing a 1 to bit[n] (n = 0..17) will clear an interrupt request to processor[n]. Writing a 0 has no effect. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of all of the processor interrupt lines.

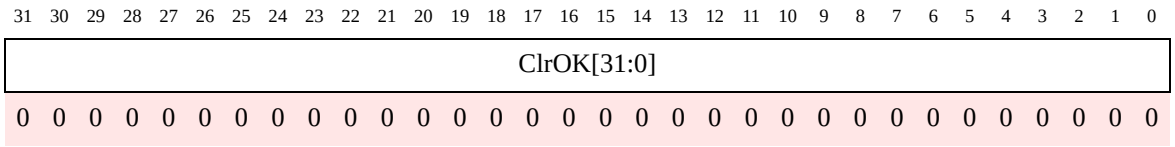
r4: Set CPU OK



Writing a 1 to bit[n] (n = 0..31) will set that bit, indicating that processor[n] is believed to be functional. Writing a 0 has no effect. Reading from this register returns the current status of all of the processor OK bits. Any bits that do not correspond to a processor number can be used for any purpose - the functions of this register are entirely defined by software.

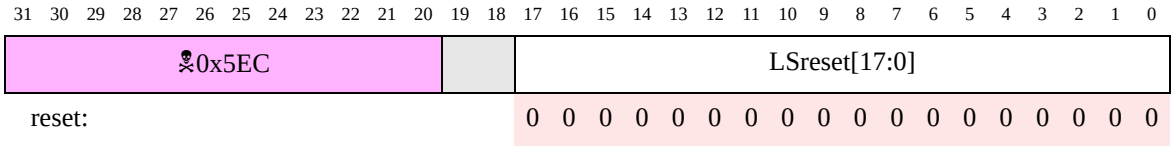
In normal use a processor will set its own bit after performing some functional self-testing. The Monitor Processor will read the register after the start-up phase to establish which processors are functional, and assign them tasks accordingly. The MP may attempt to restart faulty processors by resetting them via r6-11, or it may take them off-line by disabling their clocks via r1.

r5: Clear CPU OK



Writing a 1 to bit[n] (n = 0..31) will clear that bit, indicating that processor[n] is not confirmed as functional or has detected a fault. Writing a 0 has no effect. Reading from this register returns the current status of all of the processor OK bits.

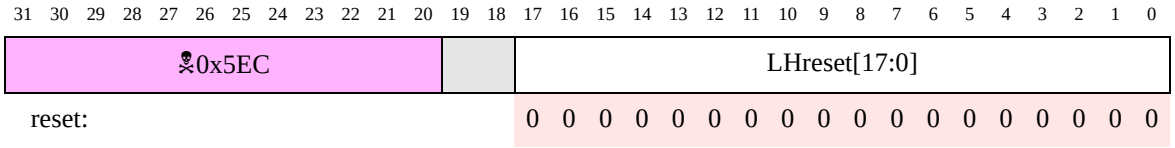
r6: CPU node soft reset - level



Writing to bit[n] (n = 0..17) will set a level on the reset input of processor[n] which is ORed with the corresponding output of the pulse reset generator, r9. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of this register, that is the level before the OR with the pulse reset output.

This is a soft reset which resets the ARM9 processor core, thereby restarting its execution at the reset vector, and resets the Communication and DMA Controllers once active transactions have completed.

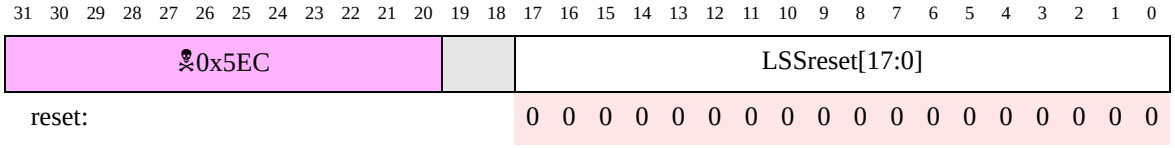
r7: CPU node hard reset - level



Writing to bit[n] (n = 0..17) will set a level on the reset input of processor node[n] which is ORed with the corresponding output of the pulse reset generator, r10. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of this register, that is the level before the OR with the pulse reset output.

This is a hard reset which resets the entire ARM968 processor node, including the peripheral hardware components in that node.

r8: Subsystem reset - level



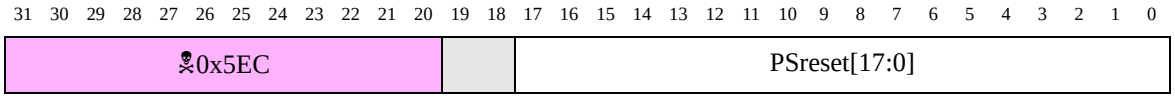
Writing a 1 to bit[n] (n = 0..17) will set a level on the reset input of a subsystem. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of this register, that is the level before the OR with the pulse reset output.

The assignment of these bits to subsystems is given in the following table:

LSSreset	Reset target
0	Router
1	PL340 SDRAM controller
2	System NoC
3	Communications NoC
4-9	Tx link 0-5

LSSreset	Reset target
10-15	Rx link 0-5
16	System AHB & Clock Gen (pulse reset only)
17	Entire chip (pulse reset only)
18-19	unassigned

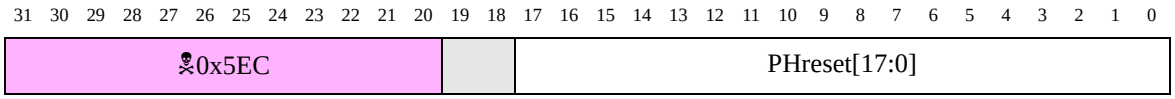
r9: CPU node soft reset - pulse



Writing a 1 to bit[n] (n = 0..17) will generate a pulse (of 256 System Controller clock cycles) on the reset input of processor[n], which is ORed with the corresponding output of the reset level register r6. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of the reset lines after the OR with the level reset output.

The reset function is as described for r6.

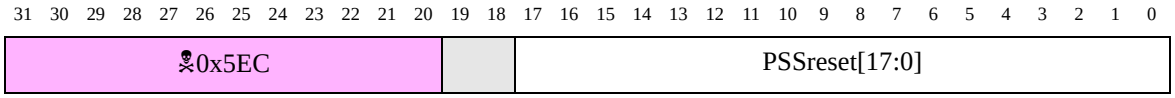
r10: CPU node hard reset - pulse



Writing a 1 to bit[n] (n = 0..17) will generate a pulse (256 clock cycles long) on the reset input of processor node[n], which is ORed with the corresponding output of the reset level register r7. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of the reset lines after the OR with the level reset output.

The reset function is as described for r7.

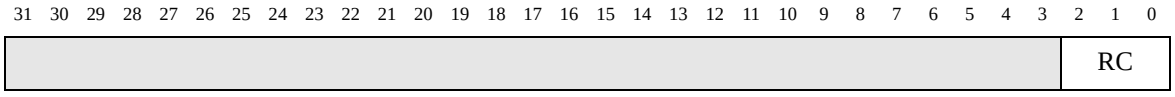
r11: Subsystem reset - pulse



Writing a 1 to bit[n] (n = 0..17) will generate a pulse (256 clock cycles long) on the reset input of a subsystem. For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX. Reading from this register returns the current status of the reset lines after the OR with the level reset output.

The assignment of these bits to subsystems is the same as that described for r8.

r12: Reset code



These bits return a code indicating the last active reset source. The reset sources are given in the

following table:

RC[2:0]	Reset source	Hard/soft reset action
000	POR - Power-on reset	hard, everything
001	WDR - Watchdog reset	hard, all but MPID[4:0] in r13
010	UR - User reset	hard, all but MPID[4:0] in r13 & B in r14
011	REC - Reset entire chip (r11 bit 17)	hard, all but MPID[4:0] in r13 & B in r14
100	WDI - Watchdog interrupt	soft, only Monitor Processor if R=1 in r13

The Power-on reset RC[2:0] = 000 hard resets everything, including setting MPID[4:0] = 11111 in r13 and B = 0 in r14.

WDR, UR and REC (RC[2:0] = 001, 010 or 011) do not reset MPID[4:0] in r13, which retains its value through the reset, thereby preventing the old Monitor Processor from competing to be Monitor Processor after the reset.

UR and REC (RC[2:0] = 010 or 011) do not reset B in r14, which will retain its value through the reset, thereby allowing booting from RAM.

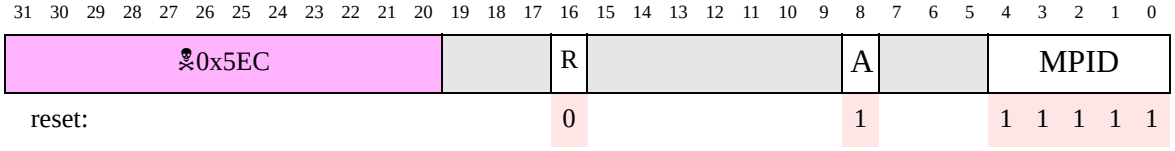
The Watchdog interrupt RC[2:0] = 100 only soft resets the Monitor Processor (with a 256 cycle pulse), and then only if this is enabled in r13.

r13: Monitor ID

This register holds the ID of the processor which has been chosen as the Monitor Processor, together with associated control bits.

Software must set the MPID value in the Router Control Register, which the Router uses to route P2P and NN packets to the Monitor Processor, to match MPID[4:0].

MPID[4:0] is initialised by power-on reset to an invalid value which does not refer to any processor. Other forms of reset do not change MPID[4:0]. It is set to the ID of the processor that wins the competition at start-up by reading its respective register r32 to r63 first.



The functions of these fields are described in the table below:

Name	bits	R/W	Function
R	16	R/W	Reset Monitor Processor on Watchdog interrupt
A	8	R/W	Write 1 to set MP arbitration bit (see r32-63)
MPID[4:0]	4:0	R/W	Monitor Processor ID

The ‘R’ bit causes the Watchdog interrupt signal to cause a soft reset of processor[MPID], which will override any interrupt masking by the Monitor Processor. In any case, this interrupt is available at all processor VICs and can therefore be enabled locally as an IRQ or FIQ source.

Reading bit[8] returns the current value of the MP arbitration bit (see r32-63).

For a write to r13 to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX.

r14: Misc control

This register supports general chip control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											R	D	C	E	T	J															B
reset:											0	0	0	0	0	0															0

The function of these fields is described in the table below:

Name	bits	R/W	Function
R	20	R	read value on JTAG_RTCK pin
D	19	R	read value on JTAG_TDO pin
C	18	R	read value on Clk32 pin
E	17	R	read value on Ethermux pin
T	16	R	read value on Test pin
J	15	R/W	select on-chip (1) or off-chip (0) control of JTAG pins
B	0	R/W	map System ROM (0) or RAM (1) to Boot area

The JTAG port is controllable by software using r14 and r16. Bit[15] of r14 selects this option when high. When selected, the GPIO bits in r16 control the JTAG inputs: GPIO[27:24] drive JTAG_NTRST, JTAG_TMS, JTAG_TDI and JTAG_TCK respectively, and the JTAG outputs JTAG_TDO and JTAG_RTCK are readable via r14 as above.

When JTAG is being driven externally, reading the r14 bits[20:19] and r16 bits[27:24] returns the state of the JTAG pins.

B is reset by power-on reset (POR) and watchdog reset (WDR).

r15: GPIO pull up/down control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO port pull up/down enable																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The functions of these bit fields are described in the table below:

bits	R/W	Function
31:29	R/W	GPIO[31:29] - on-package SDRAM control - pull-down
28:24	R/W	Unused
23:20	R/W	GPIO[23:20] & MII TxD port pull-down
19:16	R/W	GPIO[19:16] & MII RxD port pull-up
15:0	R/W	GPIO[15:0] pull-down

r16: IO port

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO port data																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register holds a 32-bit value, most bits of which may be driven out through pins when the corresponding bit in r17 is 0. When read, the values in this register are returned. The number of physical IO pins available depends on whether or not the Ethernet interface is in use. The external EtherMux input, if driven high, enables the Ethernet Tx_D[3:0] and Rx_D[3:0] onto the pins used for IO[23:16]. If EtherMux is low these pins are available for general-purpose IO use.

The functions of these bit fields are described in the table below:

bits	R/W	Function
31:29	R/W	On-package SDRAM control
28	R/W	Unused
27:24	R/W	Can drive the JTAG interface
23:20	R/W	IO pins or MII TxD
19:16	R/W	IO pins or MII RxD
15:0	R/W	IO pins

r17: IO direction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO port direction																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This register determines whether each IO port bit is an input (1) or an output (0). Setting a bit to an input does not invalidate the corresponding bit in r16 - that value will be held in r16 until explicitly changed by a write to r16. When read, this register returns the value last written.

r18: Set IO

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SetIO																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writing a 1 sets the corresponding bit in r16. Writing a 0 has no effect.

Reading this register returns the values on the IO pins (if present).

r19: Clear IO

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ClearIO																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writing a 1 clears the corresponding bit in r16. Writing a 0 has no effect.

Reading this register returns the values on the IO pins (if present).

r20: PLL1 control, and register 21: PLL2 control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							T						P	FR				MS[5:0]								NS[5:0]					
reset:							0						0	1	0			0	0	0	0	0	1			0	0	1	0	1	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
T	24	R/W	test (=0 for normal operation)
P	18	R/W	NOT power down
FR[1:0]	17:16	R/W	frequency range (10 = 100 to 200MHz)
MS[5:0]	13:8	R/W	output clock divider
NS[5:0]	5:0	R/W	input clock multiplier

The PLL output clock frequency, with a 10 MHz input clock, is given by 10*NS/MS. Thus setting NS[5:0] = 010100 [=20] and MS[5:0] = 000001 [=1] will give 200 MHz.

r22: Set flags

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit flags register																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writing a 1 to any bit position sets the corresponding bit in the flags register. Writing a 0 has no effect. Reading returns the value of the flags register.

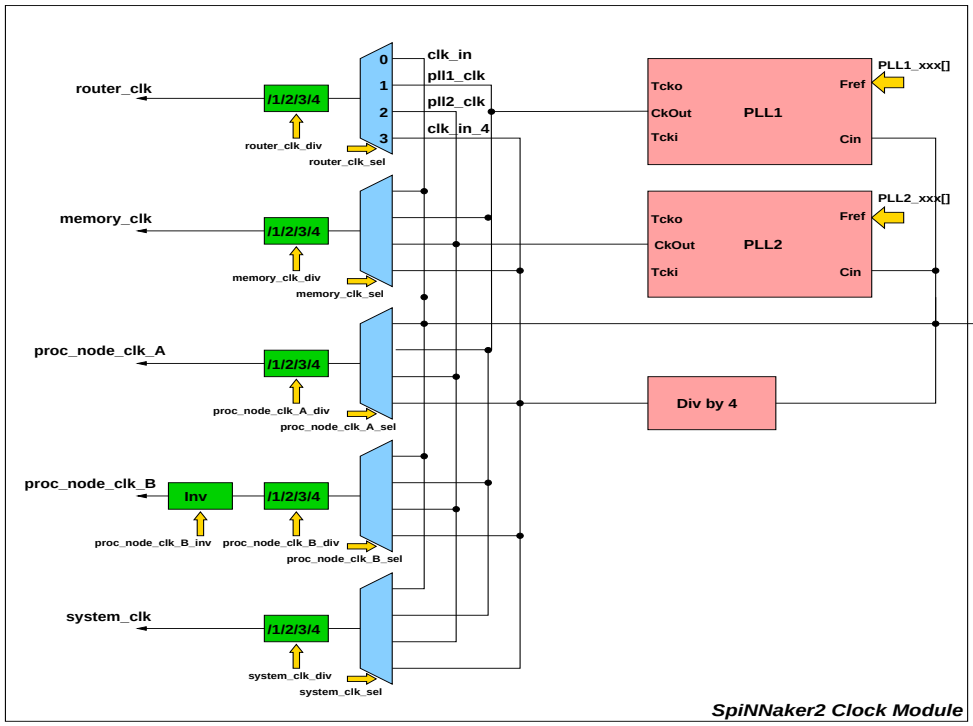
r23: Reset flags

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32-bit flags register																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writing a 1 to any bit position clears the corresponding bit in the flags register. Writing a 0 has no effect. Reading returns the value of the flags register.

r24: Clock multiplexer control

The clock generator circuits are organised as shown below:



The clock circuits are reset by r8 & r11 bit[16] and are controlled by r24:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V								Sdiv		Sys			Rdiv		Rtr			Mdiv		Mem			Bdiv		Pb			Adiv		Pa	
0								0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0		0	0	0	0

The functions of these fields are described in the table below:

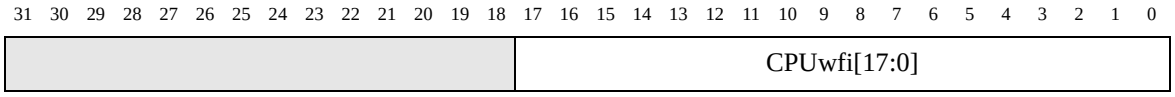
Name	bits	R/W	Function
V	31	R/W	invert CPU clock B
Sdiv[1:0]	23:22	R/W	divide System AHB clock by Sdiv+1 (= 1-4)
Sys[1:0]	21:20	R/W	clock selector for System AHB components
Rdiv[1:0]	18:17	R/W	divide Router clock by Rdiv+1 (= 1-4)
Rtr[1:0]	16:15	R/W	clock selector for Router
Mdiv[1:0]	13:12	R/W	divide SDRAM clock by Mdiv+1 (= 1-4)
Mem[1:0]	11:10	R/W	clock selector for SDRAM
Bdiv[1:0]	8:7	R/W	divide CPU clock B by Bdiv+1 (= 1-4)
Pb[1:0]	6:5	R/W	clock selector for B CPUs (0 3 5 6 9 10 12 15 17)
Adiv[1:0]	3:2	R/W	divide CPU clock A by Adiv+1 (= 1-4)
Pa[1:0]	1:0	R/W	clock selector for A CPUs (1 2 4 7 8 11 13 14 16)

All clock selectors choose from the same clock sources:

Sel[1:0]	Clock source
00	external 10MHz clock input
01	PLL1
10	PLL2
11	external 10MHz clock divided by 4

Clock switching is safe at any time once the PLLs have locked, which takes a defined time (maximum 80µs for the PLLs) after they have been configured.

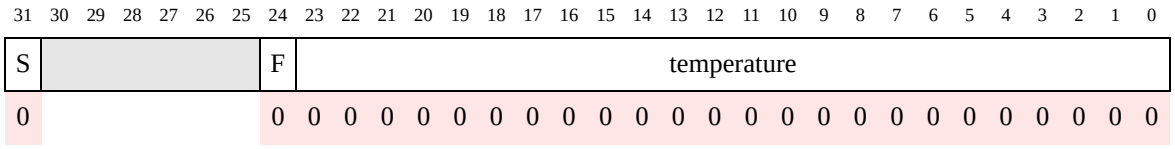
r25: CPU sleep status



Each bit in this register indicates the state of the respective ARM968 STANDBYWFI (stand-by wait for interrupt) signal, which is active when the CPU is in its low-power sleep mode.

r26-28: Temperature sensor registers

There are three independent temperature sensors on the chip, each with its own control and sensor read-out register. The three sensors use different sensor mechanisms to enable the temperature to be corrected for process and voltage variations.

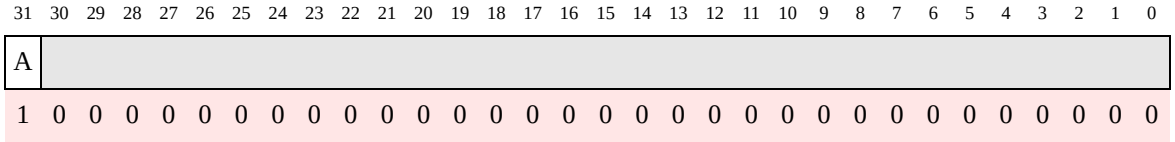


The functions of these fields are described in the table below:

Name	bits	R/W	Function
S	31	R/W	start temperature measurement
F	24	R	temperature measurement finished
temperature	23:0	R	temperature sensor reading

Setting S to 1 starts the temperature measurement process. When F reads as 1 the sensor reading is complete, and bits[23:0] may be read. Clearing S stops the sensing and clears F.

r32-63: Monitor Processor arbitration

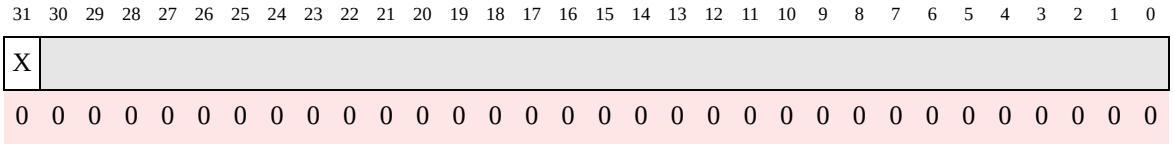


The same single-bit value ‘A’ appears in all registers r32 to r63.

'A' is set by a reset event (with RC[1:0] = 000, 001, 010 or 011 in r12) and can also be set by software via r13 bit[8]. A processor which has passed its self-test may read this register at address offset 0x80 + 4*N, where N is the processor's number. If A is set when the read takes place and N is not equal to the current value in r13 (the Monitor Processor ID register), 0x80000000 is returned, N is placed in r13, and A is cleared.

If A is clear when the read takes place, or N equals the current value in r13, then the value 0x00000000 is returned and A and r13 are unchanged.

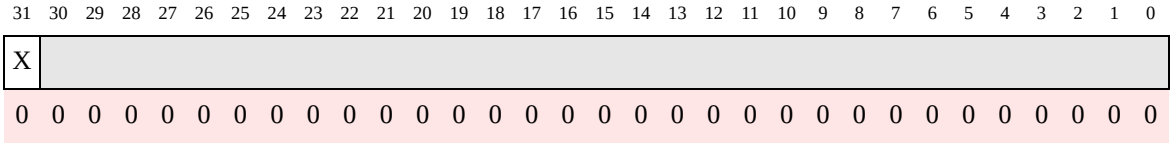
r64-95: Test and Set



A unique single-bit value 'X' appears in each register r64 to r95. Reading each register returns 0x00000000 or 0x80000000 depending on whether its respective bit was clear or set prior to the read, and as a side-effect the bit is set by the read.

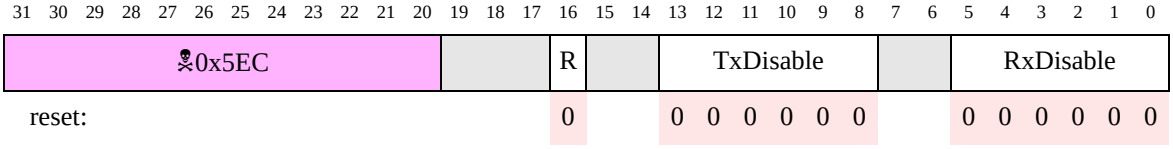
Together with r96 to r127, these registers provide support for mutual exclusion primitives for inter-processor communication and shared data structures, compensating for the lack of support for locked ARM 'swap' instructions into the System RAM.

r96-127: Test and Clear



The same unique single-bit value 'X' appears in each register r96 to r127 as appears in r64 to r95 respectively. Reading each register returns 0x00000000 or 0x80000000 depending on whether its respective bit was clear or set prior to the read, and as a side-effect the bit is cleared by the read.

r128: Tx and Rx link disable



For a write to be effective it must include a security code in bits [31:20]: 0x5ECXXXXX.

The functions of these fields are described in the table below:

Name	bits	R/W	Function
R	16	R/W	Router parity control
TxDisable[5:0]	13:8	R/W	disables the corresponding link transmitter
RxDisable[5:0]	5:0	R/W	disables the corresponding link receiver

15. Ethernet MII interface

The SpiNNaker system connects to a host machine via Ethernet links. Each SpiNNaker chip includes an Ethernet MII interface, although only a few of the chips are expected to use this interface. These chips will require an external PHY.

The interface hardware operates at the frame level. All higher-level protocols will be implemented in software running on the local monitor processor.

15.1 Features

- support for full-duplex 10 and 100 Mbit/s Ethernet via off-chip PHY
- outgoing 1.5Kbyte frame buffer, for one maximum-size frame
  - outgoing frame control, CRC generation and inter-frame gap insertion
- incoming 3Kbyte frame buffer, for two maximum-size frames
  - incoming frame descriptor buffer, for up to 48 frame descriptors
  - incoming frame control with length and CRC check
  - support for unicast (with programmable MAC address), multicast, broadcast and promiscuous frame capture
  - receive error filter
- internal loop-back for test purposes
- general-purpose IO for PHY management (SMI) and PHY reset
- interrupt sources for frame-received, frame-transmitted and PHY (external) interrupt

[The implementation does not provide support for half-duplex operation (as required by a CSMA/CD MAC algorithm), jumbo or VLAN frames.]

15.2 Using the Ethernet MII interface

The Ethernet driver software must observe a number of sequence dependencies in initialising the PHY and setting-up the MAC address before the Ethernet interface is ready for use.

Details of these issues are documented in “SpiNNaker AHB-MII module” by Brendan Lynskey. The latest version of this is v003, February 2008.

15.3 Register summary

Base address: 0xe4000000 (buffered write), 0xf4000000 (unbuffered write).

User registers

The following registers allow normal user programming of the Ethernet interface:

Name	Offset	R/W	Function
Tx frame buffer	0x0000	W	Transmit frame RAM area
Rx frame buffer	0x4000	R	Receive frame RAM area
Rx desc RAM	0x8000	R	Receive descriptor RAM area
r0: Gen command	0xC000	R/W	General command
r1: Gen status	0xC004	R	General status

Name	Offset	R/W	Function
r2: Tx length	0xC008	R/W	Transmit frame length
r3: Tx command	0xC00C	W	Transmit command
r4: Rx command	0xC010	W	Receive command
r5: MAC addr ls	0xC014	R/W	MAC address low bytes
r6: MAC addr hs	0xC018	R/W	MAC address high bytes
r7: PHY control	0xC01C	R/W	PHY control
r8: Interrupt clear	0xC020	W	Interrupt clear
r9: Rx buf rd ptr	0xC024	R	Receive frame buffer read pointer
r10: Rx buf wr ptr	0xC028	R	Receive frame buffer write pointer
r11: Rx dsc rd ptr	0xC02C	R	Receive descriptor read pointer
r12: Rx dsc wr ptr	0xC030	R	Receive descriptor write pointer

Test registers

In addition, there are test registers that will not normally be of interest to the programmer:

Name	Offset	R/W	Function
r13: Rx Sys state	0xC034	R	Receive system FSM state (debug & test use)
r14: Tx MII state	0xC038	R	Transmit MII FSM state (debug & test use)
r15: PeriphID	0xC03C	R	Peripheral ID (debug & test use)

See “SpiNNaker AHB-MII module” by Brendan Lynskey version 003, February 2008 for further details of the test registers.

15.4 Register details

r0: General command register

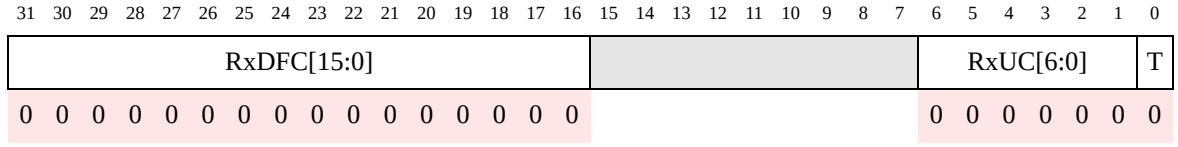
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					H	D	V	P	B	M	U	F	L	R	T
reset:																					0	0	0	0	1	1	1	1	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
H	10	R/W	Disable hardware byte reordering
D	9	R/W	Reset receive dropped frame count (in r1)
V	8	R/W	Receive VLAN enable

Name	bits	R/W	Function
P	7	R/W	Receive promiscuous packets enable
B	6	R/W	Receive broadcast packets enable
M	5	R/W	Receive multicast packets enable
U	4	R/W	Receive unicast packets enable
F	3	R/W	Receive error filter enable
L	2	R/W	Loopback enable
R	1	R/W	Receive system enable
T	0	R/W	Transmit system enable

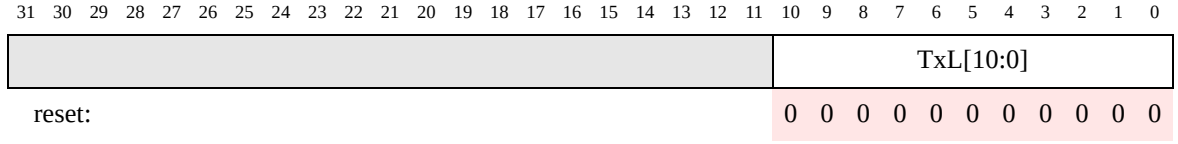
r1: General status register



The functions of these fields are described in the table below:

Name	bits	R/W	Function
RxDFC[15:0]	31:16	R	Receive dropped frame count
RxUC[5:0]	7:1	R	Received unread frame count
T	0	R	Transmit MII interface active

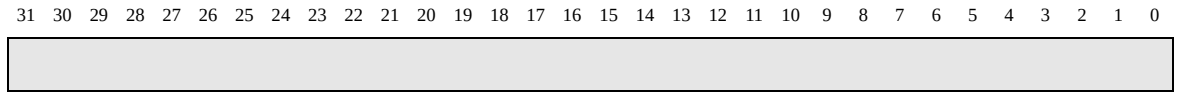
r2: Transmit frame length



The functions of these fields are described in the table below:

Name	bits	R/W	Function
TxL[10:0]	10:0	R/W	Length of transmit frame (60 - 1514 bytes)

r3: Transmit command register



Any write to register 3 causes the transmission of a frame.

r4: Receive command register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Any write to register 4 indicates that the current receive frame has been processed and decrements the received unread frame count in register 1.

r5: MAC address low bytes

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC3[7:0]								MAC2[7:0]								MAC1[7:0]								MAC0[7:0]							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
MAC3[7:0]	31:24	R/W	MAC address byte 3
MAC2[7:0]	23:16	R/W	MAC address byte 2
MAC1[7:0]	15:8	R/W	MAC address byte 1
MAC0[7:0]	7:0	R/W	MAC address byte 0

r6: MAC address high bytes

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																MAC5[7:0]								MAC4[7:0]							
reset:																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
MAC5[7:0]	15:8	R/W	MAC address byte 5
MAC4[7:0]	7:0	R/W	MAC address byte 4

r7: PHY control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										Q	C	E	O	I	R
reset:																										0	0	0	0		0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
Q	5	R/W	PHY IRQn invert disable
C	4	R/W	SMI clock (active rising)
E	3	R/W	SMI data output enable
O	2	R/W	SMI data output
I	1	R	SMI data input
R	0	R/W	PHY reset (active low)

r8: Interrupt clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											R				T

The functions of these fields are described in the table below:

Name	bits	R/W	Function
R	4	W	Clear receive interrupt request
T	0	W	Clear transmit interrupt request

Writing a 1 to bit [0] if this register clears a pending transmit frame interrupts. Writing a 1 to bit [4] clears a pending receive frame interrupt. There is no requirement to write a 0 to these bits other than in order to prevent unintentional interrupt clearance.

r9: Receive frame buffer read pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			V	RFBRP[11:0]											
reset:																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
V	12	R	Rollover bit - toggles on address wrap-around
RFBRP[11:0]	11:0	R	Receive frame buffer read pointer

r10: Receive frame buffer write pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			V	RFBWP[11:0]											
reset:																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
V	12	R	Rollover bit - toggles on address wrap-around
RFBWP[11:0]	11:0	R	Receive frame buffer write pointer

r11: Receive descriptor read pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									V	RDRP[5:0]					
reset																									0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
V	6	R	Rollover bit - toggles on address wrap-around
RDRP[5:0]	5:0	R	Receive descriptor read pointer

r12: Receive descriptor write pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									V	RDWP[5:0]					
reset																									0	0	0	0	0	0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
V	6	R	Rollover bit - toggles on address wrap-around
RDWP[5:0]	5:0	R	Receive descriptor write pointer

15.5 Fault-tolerance

The Ethernet interface will only be used on a small number of nodes; most nodes are insensitive to faults in its functionality as they will not attempt to use it.

16. Watchdog timer

The watchdog timer is an ARM PrimeCell component (ARM part SP805, documented in ARM DDI 0270B) that is responsible for applying a system reset when a failure condition is detected. Normally, the Monitor Processor will be responsible for resetting the watchdog periodically to indicate that all is well. If the Monitor Processor should crash, or fail to reset the watchdog during a pre-determined period of time, the watchdog will trigger.

16.1 Features

- generates an interrupt request after a programmable time period;
- causes a chip-level reset if the Monitor Processor does not respond to an interrupt request within a subsequent time period of the same length.

16.2 Register summary

Base address: 0xe3000000 (buffered write), 0xf3000000 (unbuffered write).

User registers

The following registers allow normal user programming of the Watchdog timer:

Name	Offset	R/W	Function
r0: WdogLoad	0x00	R/W	Count load register
r1: WdogValue	0x04	R	Current count value
r2: WdogControl	0x08	R/W	Control register
r3: WdogIntClr	0x0C	W	Interrupt clear register
r4: WdogRIS	0x10	R	Raw interrupt status register
r5: WdogMIS	0x14	R	Masked interrupt status register
r6: WdogLock	0xC00	R/W	Lock register

Test and ID registers

In addition, there are test and ID registers that will not normally be of interest to the programmer:

Name	Offset	R/W	Function
WdogITCR	0xF00	R/W	Watchdog integration test control register
WdogITOP	0xF04	W	Watchdog integration test output set register
WdogPeriphID0-3	0xFE0-C	R	Watchdog peripheral ID byte registers
WdogPCID0-3	0xFF0-C	R	Watchdog Prime Cell ID byte registers

See AMBA Design Kit Technical Reference Manual ARM DDI 0243A, February 2003, for further details of the test and ID registers.

16.3 Register details

r0: Load

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wdog load																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This read-write register contains the value the from which the counter is to decrement. When this register is written to, the count immediately restarts from the new value. The minimum value is 1.

r1: Count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wdog count																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This read-only register contains the current value of the decrementing counter. The first time the counter decrements to zero the Watchdog raises an interrupt. If the interrupt is still active the second time the counter decrements to zero the reset output is activated.

r2: Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														E	I
reset:																														0	0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
E	1	R/W	Enable the Watchdog reset output (1)
I	0	R/W	Enable Watchdog counter and interrupt (1)

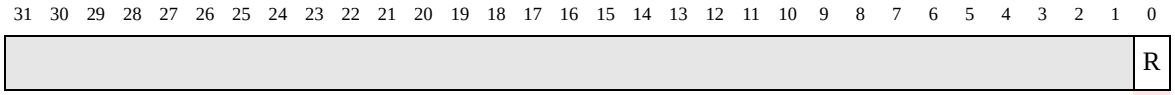
Once the Watchdog has been initialised both enables should be set to ‘1’ for normal watchdog operation.

r3: Interrupt clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

A write of any value to this register clears the watchdog interrupt and reloads the counter from r1.

r4: Raw interrupt status

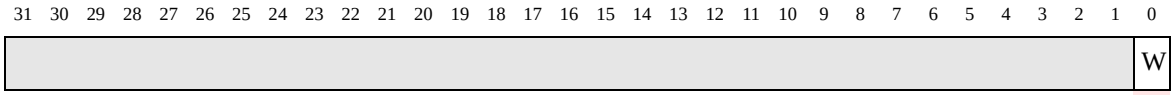


reset: 0

The function of this field is described in the table below:

Name	bits	R/W	Function
R	0	R	Raw (unmasked) watchdog interrupt

r5: Masked interrupt status

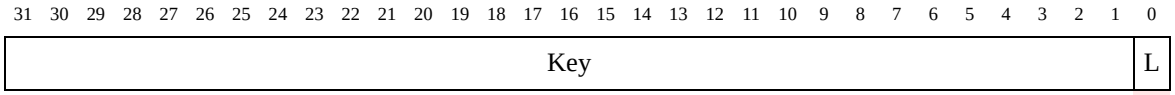


reset: 0

The function of this field is described in the table below:

Name	bits	R/W	Function
W	0	R	Watchdog interrupt output

r6: Lock



reset: 0

The functions of these fields are described in the table below:

Name	bits	R/W	Function
Key	31:0	W	Write 0x1ACCE551 to enable writes
L	0	R	Write access enabled (0) or disabled (1)

A read from this register returns only the bottom bit, indicating whether writes to other registers are enabled (0) or disabled (1). A write of 0x1ACCE551 enables write access to the other registers; a write of any other value disables write access to the other registers. Note that the 'Key' field is 32 bits and includes bit 0.

The lock function is available to ensure that the watchdog will not be reset by errant programs.

17. System RAM

The System RAM is an additional *32* Kbyte block of on-chip RAM used primarily by the Monitor Processor to enhance its program and data memory resources as it will be running more complex (though less time-critical) algorithms than the fascicle processors.

As the choice of Monitor Processor is made at start-up (and may change during run-time for fault-tolerance purposes) the System RAM is made available to whichever processor is Monitor Processor via the System NoC. Accesses by the Monitor Processor to the System RAM are non-blocking as far as SDRAM accesses by the fascicle processors are concerned.

The System RAM may also be used by the fascicle processors to communicate with the Monitor Processor and with each other, should the need arise.

17.1 Features

- *32* Kbytes of SRAM, available via the System NoC.
- can be used as source of boot code.

17.2 Address location

Base address: 0xe5000000 (buffered write), 0xf5000000 (unbuffered write). Can also appear at the Boot area at 0xff000000 if the 'Boot area switch' is set in the System Controller.

17.3 Fault-tolerance

Fault insertion

- It is straightforward to corrupt the contents of the System RAM to model a soft error – any processor can do this. It is not clear how this would be detected.

Fault detection

- The Monitor Processor may perform a System RAM test at start-up, and periodically thereafter.
- It is not clear how soft errors can be detected without some sort of parity or ECC system.

Fault isolation

- Faulty words in the System SRAM can be mapped out of use.

Reconfiguration

- For hard failure of a single bit, avoid using the word containing the failed bit.
- If the System RAM fails completely the only option is to use the SDRAM instead, which will probably result in compromised performance for the fascicle processors due to loss of SDRAM bandwidth. An option then would be to relocate some of the fascicle processors' workload to another chip.

17.4 Test

Production test

- run standard memory test patterns from one of the processing subsystems.

18. Boot ROM

18.1 Features

- a small (32Kbyte) on-chip ROM to provide minimal support for:
  - initial self-test, and Monitor Processor selection
  - Router initialisation for bootstrapping
  - system boot.

  The Test chip Boot ROM also supports the loading of code from an external SPI ROM using the GPIO[5:2] pins as an SPI interface.

18.2 Address location

Base address: 0xf6000000 and, after a hard reset and unless the 'Boot area switch' is set in the Sytem Controller, in the Boot area at 0xff000000.

18.3 Fault-tolerance

Fault insertion

Switch the 'Boot area switch' to remove the Boot ROM from the reset location.

Fault detection

If the Boot ROM fails the boot process will also fail, which will be detected at start-up.

Fault isolation

Switching the Boot ROM out of the boot area should render it harmless.

Reconfiguration

When the Boot ROM is switched out of the boot area the System RAM is switched into the boot area. A neighbour 'nurse' chip can initialise the System RAM with the boot code and retry initialisation.

19. JTAG

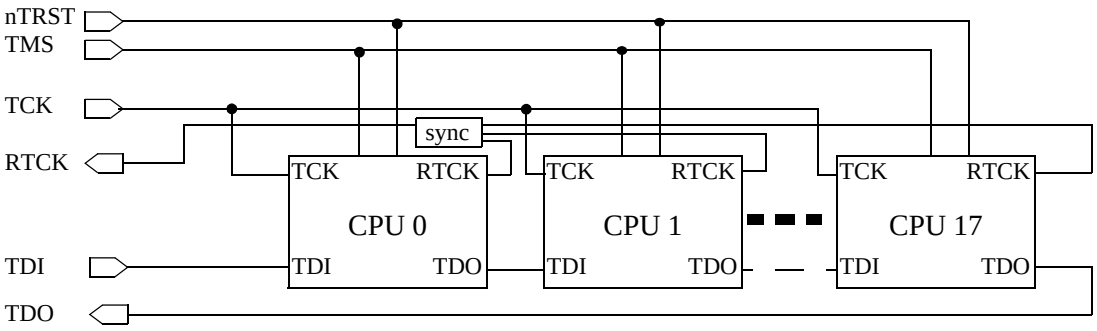
The JTAG IEEE 1149.1 system on the SpiNNaker chip provides access only to the ARM968 processors for software debug purposes. There is no provision for scan access to the SpiNNaker pins or other on-chip features.

19.1 Features

- standard ARM debug access to all *18* ARM968 processors
- device ID codes of 0x05968477

19.2 Organisation

The organisation of the ARM968 JTAG access is as shown below:



The ARM968 CPUs synchronize TCK to their respective local clocks, which may be different, so the ARM interface has an addition clock return signal, RTCK, which indicates when a transition on TCK has been recognised. TCK may then make a further transition. The RTCK signal allows TCK to be operated at the maximum safe frequency.

TCK and RTCK should obey a standard handshake protocol, so TCK may only rise when RTCK is low, and TCK may only fall when RTCK is high.

All of the processors are in series on the data scan path (TDI to TDO), with CPU0 coming before CPU1, etc. All processor TAP controllers have JTAG-standard bypass registers to support more efficient access to the other processor.

19.3 Operation

The JTAG interface supports direct connection of the ARM software development tools to the SpiNNaker test chip, giving those tools standard access to the ARM processors, their local memories, and all system functions visible from those processors.

It is expected that the JTAG interface will be used only with suitable JTAG-aware tools, for hardware debugging (if necessary) and software debugging as required.

20. Input and Output signals

The SpiNNaker chip has the following IO, power and ground pins. All IO is assumed to operate at 1.8V with CMOS logic levels; the SDRAM interface is 1.8V LVCMOS. All other IOs are non-critical, though output delay affects link throughput.

20.1 Key

The 'Drive' column in the tables uses the following notation:

Direction	Drive	Meaning
output	*N*mA	maximum drive current *N* mA
output	A/B	slow/fast slew rate
input	S	Schmitt trigger input
input	D/U	pull down/up resistor incorporated

20.2 SDRAM interface

Signal	Type	Drive	Function	#
DQ[31:0]	IO	8mA B	Data	1-32
A[13:0]	O	4mA B	Address	33-46
CK, CK#	O	8mA B	True and inverse clock	47, 48
CKE	O	4mA B	Clock enable	49
CS#[1:0]	O	4mA B	Chip selects	50, 51
RAS#	O	4mA B	Row address strobe	52
CAS#	O	4mA B	Column address strobe	53
WE#	O	4mA B	Write enable	54
DM[3:0]	O	8mA B	Data mask	55-58
BA[1:0]	O	4mA B	Bank address	59, 60
DQS[3:0]	IO	8mA DB	Data strobe	61-64
Vdd_18[23, 13:0]	1.8V		Power for SDRAM pins	65-79
Vss_18[23, 13:0]	Gnd		Ground for SDRAM pins	80-94

When the package incorporates an internal SDRAM die, all of the above signal pins apart from CS#[1] will be connected to it. They may or may not also be connected to package balls. CS#[1] connects only to a package ball.

20.3 JTAG

Signal	Type	Drive	Function	#
nTRST	I	SU	Test reset (active low)	95
TCK	I	SD	Test clock	96
RTCK	O	4mA A	Return test clock	97
TMS	I	SU	Test mode select	98
TDI	I	SU	Test data in	99
TDO	O	4mA A	Test data out	100

20.4 Ethernet MII

Signal	Type	Drive	Function	#
EtherMux	I	SD	select Ethernet or GPIO[23:16]	101
RX_CLK	I	SD	Receive clock	102
RX_D[3:0]	IO	4mA A SU	Receive data/GPIO[19:16]	103-106
RX_DV	I	SD	Receive data valid	107
RX_ERR	I	SD	Receive data error	108
TX_CLK	O	4mA A	Transmit clock	109
TX_D[3:0]	IO	4mA A SD	Transmit data/GPIO[23:20]	110-113
TX_EN	O	4mA A	Transmit data valid	114
TX_ERR	O	4mA A	Force transmit data error	115
MDC	O	4mA A	Management interface clock	116
MDIO	IO	4mA A	Management interface data	117
PHY_RSTn	O	4mA A	PHY reset (optional)	118
PHY_IRQn	I	SD	PHY interrupt (optional)	119
Vdd_18[15]	1.8V		Power for Ethernet MII pins	120
Vss_18[15]	Gnd		Ground for Ethernet MII pins	121

20.5 Communication links

Signal	Type	Drive	Function	#
L0in[6:0]	I	SD	link 0 2-of-7 input code	122-128
L0inA	O	12mA B	link 0 input acknowledge	129
L0out[6:0]	O	12mA B	link 0 2-of-7 output code	130-136
L0outA	I	SD	link 0 output acknowledge	137
L1in[6:0]	I	SD	link 1 2-of-7 input code	138-144
L1inA	O	12mA B	link 1 input acknowledge	145
L1out[6:0]	O	12mA B	link 1 2-of-7 output code	146-152
L1outA	I	SD	link 1 output acknowledge	153
L2in[6:0]	I	SD	link 2 2-of-7 input code	154-160
L2inA	O	12mA B	link 2 input acknowledge	161
L2out[6:0]	O	12mA B	link 2 2-of-7 output code	162-168
L2outA	I	SD	link 2 output acknowledge	169
L3in[6:0]	I	SD	link 3 2-of-7 input code	170-176
L3inA	O	12mA B	link 3 input acknowledge	177
L3out[6:0]	O	12mA B	link 3 2-of-7 output code	178-184
L3outA	I	SD	link 3 output acknowledge	185
L4in[6:0]	I	SD	link 4 2-of-7 input code	186-192
L4inA	O	12mA B	link 4 input acknowledge	193
L4out[6:0]	O	12mA B	link 4 2-of-7 output code	194-200
L4outA	I	SD	link 4 output acknowledge	201
L5in[6:0]	I	SD	link 5 2-of-7 input code	202-208
L5inA	O	12mA B	link 5 input acknowledge	209
L5out[6:0]	O	12mA B	link 5 2-of-7 output code	210-216
L5outA	I	SD	link 5 output acknowledge	217
Vdd_18[22:21, 17:14]	1.8V		Power for link pins	218-223
Vss_18[22:21, 17:14]	Gnd		Ground for link pins	224-229

20.6 Miscellaneous

Signal	Type	Drive	Function	#
GPIO[15:0]	IO	4mA A SD	General-purpose IO	230-245
PORIn	I	SD	Power-on reset	246
ResetIn	I	SD	Chip reset	247
Test	I	SD	Chip test mode	248
Clk10MIn	I	S	Main input clock - 10MHz	249
nClk10MOut	O	4mA A	Daisy-chain 10MHz clock out	250
Clk32kIn	I	S	Slow (global) 32kHz clock	251
Vdd_18[18:17]	1.8V		Power for miscellaneous pins	252-253
Vss_18[18:17]	Gnd		Ground for misc. pins	254-255
Vdd_12[13:0]	1.2V		Power for core logic	256-269
Vss_12[13:0]	Gnd		Ground for core logic	270-283
Vdd_PLL[3:0]	1.2V		Power for PLLs	284-287
Vss_PLL[3:0]	Gnd		Ground for PLLs	288-291
Tres	I	analogue	Temp. sensor analogue input	292

20.7 Internal SDRAM interface

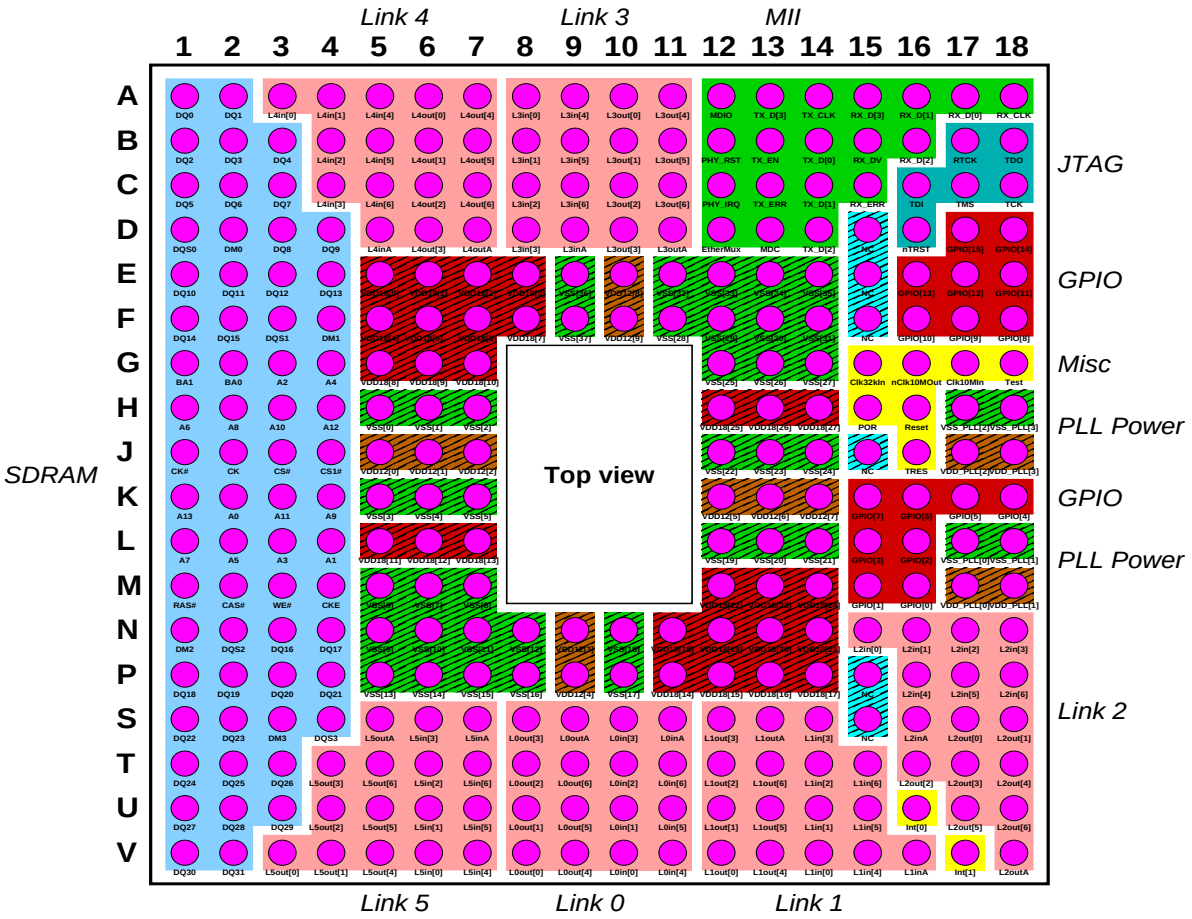
Signal	Type	Drive	Function	#
GPIO[31]	IO	4mA A SD	Connects to SDRAM TQ	293
GPIO[30]	IO	4mA A SD	SDRAM DPD input	294
GPIO[29]	IO	4mA A SD	Bond to Vdd	295

20.8 Internal SDRAM power & ground

In addition to the signal pins that connect the internal SDRAM to the SpiNNaker chip, the SDRAM also requires 1.8V Vdd and ground connections - 30 in total.

21. Packaging

The SpiNNaker chip is packaged in a 300LBGA package with 1mm ball pitch. The allocation of signals to balls is as shown below:



It is expected that a 128Mbyte Mobile DDR SDRAM will normally be incorporated into the package with the SpiNNaker chip, using wire-bonded Multi-Chip Package (MCP) assembly.

22. Application notes

22.1 Firefly synchronization

The local time phase, used for errant packet trapping, can be maintained across the system by a combination of local slightly randomized timers and local phase-locking using nearest-neighbour communication.

Time phase accuracy

If the system time phase is F and the skew is K (that is, all parts of the system transition from one phase to its successor within a time K), then a packet has at least F-K to reach its destination and will be killed after at most 2F+K. Thus, if we want to allow for a maximum packet transit time of F-K = T and can achieve a minimum phase skew of K, then T and K are both system constants and we should choose F = T+K. The longest packet life is then 2T+3K.

22.2 Neuron address space

Neurons ocuppy an address space that identifies each Neuron uniquely within the domain of its multicast routing path (where this domain must include alternative links that may be taken during emergency routing). Where these domains do not overlap it is possible to reuse the same address, though this must be done with considerable care. Neuron addresses can be assigned arbitrarily; this can be exploited to optimize Router utilization (e.g. by giving Neurons with the same routing requirements related addresses so that they can be routed by the same Router entries).