AXIS ETRAX 100LX
Designer's Reference

AXIS COMMUNICATIONS

Axis Communications AB cannot be held responsible for any technical or typographical errors, and reserves the right to make changes to this manual and to the product without prior notice. If you do detect any inaccuracies or omissions, please inform us at:

E-mail: technology@axis.com

Axis Communications AB
Emdalavägen 14
SE-223 69 Lund, Sweden
Phone:+46 46 272 1800
Fax: +46 46 13 61 30

1 Introduction

Optimized Network Controller with
RISC CPU, Cache
and Multiple I/O Ports

AXIS COMMUNICATIONS

1.1 Overview

The AXIS ETRAX 100LX is a single-chip integrated circuit designed for embedded network connectivity applications. The ETRAX 100LX improves upon the features available for the AXIS ETRAX 100, including support for Universal Serial Bus 1.1. It is compatible with the widespread ETRAX family, and offers further advances in microprocessor design and performance. The ETRAX 100LX chip incorporates the AXIS CRIS CPU which not only suits all the requirements of a network connectivity product, but also acts as an integrated core especially suited for our system.

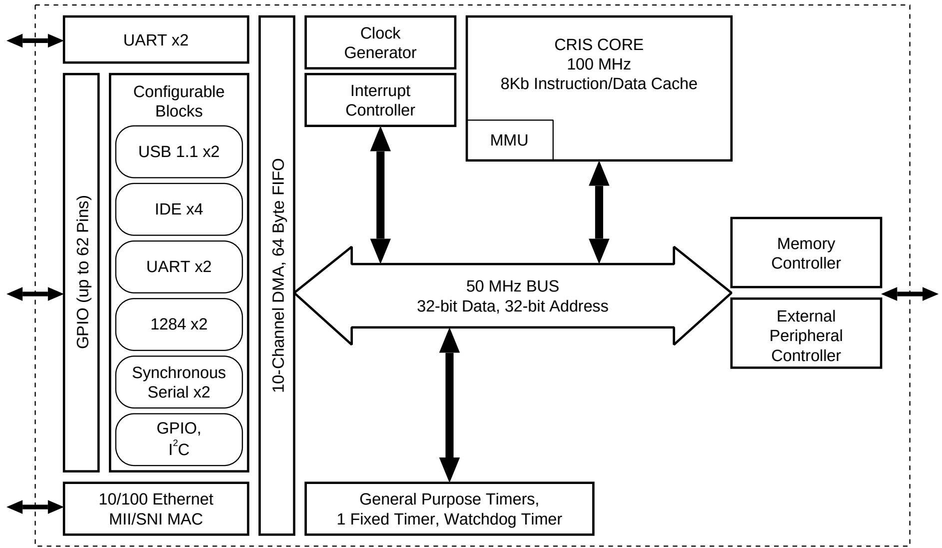
The ETRAX 100LX is ideal in executing multi protocol network stacks on one chip. The ETRAX 100LX has a 100 MIPS RISC CPU, 8 kilobyte unified instruction/data cache, high bandwidth DMA controlled I/O ports, and an on-chip Fast Ethernet controller. Its integrated functions, minimal power consumption, and high code density makes it highly suitable for a wide range of embedded applications that require high performance and low system cost.

The ETRAX 100LX programmable bus interface supports both 16-bit and 32-bit data bus widths, and interfaces directly to SDRAM, EDO DRAM, SRAM, EPROM, parallel EEPROM, and FlashPROM.

1.2 Features

- High performance 100 MIPS (200 MIPS/W) 32-bit RISC CPU, 112k Dhrystones.
- Designed specifically for running Linux by including an MMU.
- Ethernet controller supports 100Mbit/10Mbit MII (Compatible with IEEE 802.3 and Fast Ethernet standards).
- Four asynchronous serial ports with an internal baudrate programmable from 48 Hz to 6.25 MHz, and an external baudrate up to 3.125 MHz.
- Two synchronous serial ports. Master or Slave synchronous serial mode with a codec clock between 32 kHz and 4.096 MHz.
- Universal Serial Bus 1.1 Host and Device mode operation. Hardware support for dynamic connect/disconnect, suspend/resume and remote wakeup.
- Configuration of up to four EIDE/ATA-2 ports for up to 8 IDE disk drives.
- 16-bit general I/O port. The direction of each bit can be individually controlled.
- Two configurable parallel I/O ports for Centronics, IEEE 1284 byte, ECP, and EPP mode, and Shared RAM interface.
- Optimized for compact code and high speed with configurable 16-bit or 32-bit bus width.
- Bus interface supporting SDRAM, EDO DRAM, SRAM, EPROM, parallel EEPROM, and FlashPROM.
- 8 kilobyte on chip cache memory.
- DMA controlled network and port I/O for high performance
- Excellent C/C++ language support and high code density.
- Configurable bootstrap through network, serial, and parallel ports as well as FlashPROM.
- Low power consumption, 350 mW typically.
- 256-pin PBGA package, 27 x 27 x 2.15 mm.

1.3 Functional Block Diagram



The CPU in ETRAX 100LX is a RISC CPU with internal cache memory. Data handling is provided by internal DMA within the chip as well as to and from external units. The internal clocks are generated by a PLL clock multiplier that takes its input from an external clock generator. ETRAX 100LX provides internal and external vectorized interrupt.

2 RISC CPU

The CPU in ETRAX 100LX is a 32-bit RISC CPU with a 16-bit wide instruction. The CPU complies with the Axis Code Reduced Instruction Set (CRIS) architecture. It runs at a cycle frequency of 100 MHz, giving a peak performance of 100 MIPS. A summary of the CRIS architecture is given below. The CRIS CPU architecture is described in more detail in the “ETRAX 100LX Programmer’s Manual”.

2.1 Registers

The processor contains 14 32-bit *General Registers* (R0 - R13), one 32-bit *Stack Pointer* (R14 or SP), and one 32-bit *Program Counter* (R15 or PC).

The processor architecture also contains 16 *Special Registers* (P0 - P15), ten of which are implemented. The registers are presented in the figures below:

General Registers:

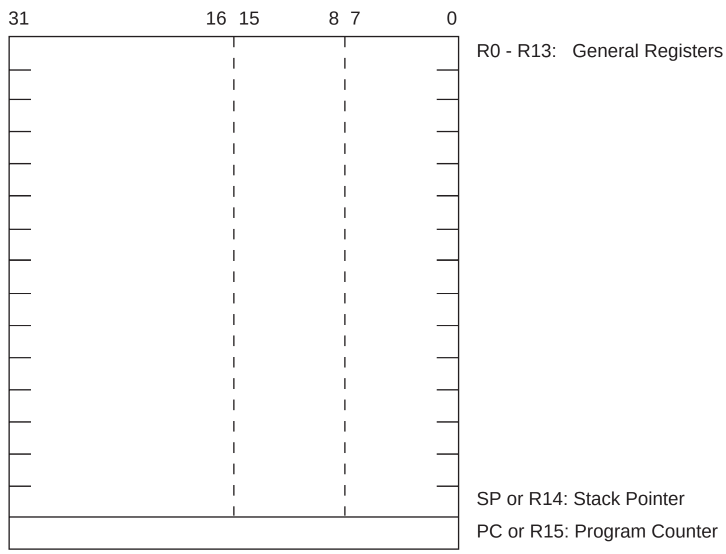


Figure 2-1 General Registers

Special Registers:

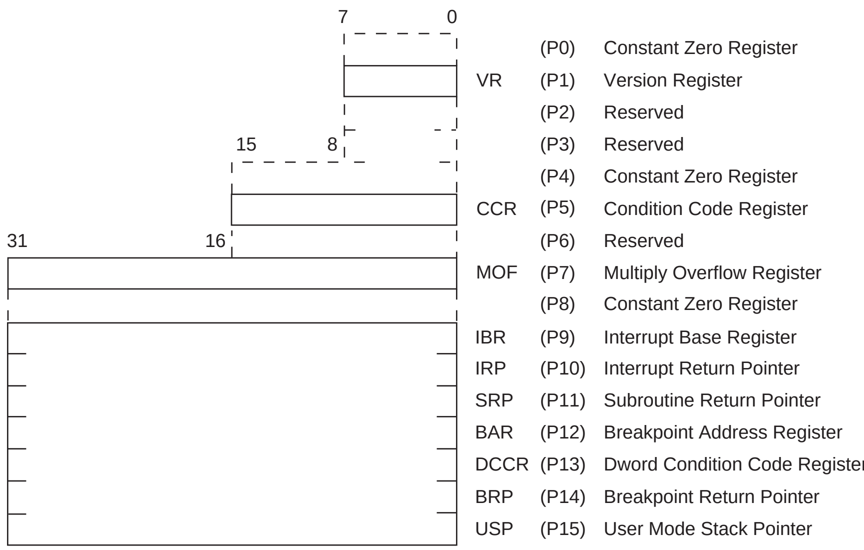


Figure 2-2 Special Registers

2.2 Flags and Condition Codes

The Condition Code Register (CCR) and its 32-bit extension, the Dword Condition Code Register (DCCR), for the ETRAX 100LX contain eleven different flags. The remaining bits are always zero:

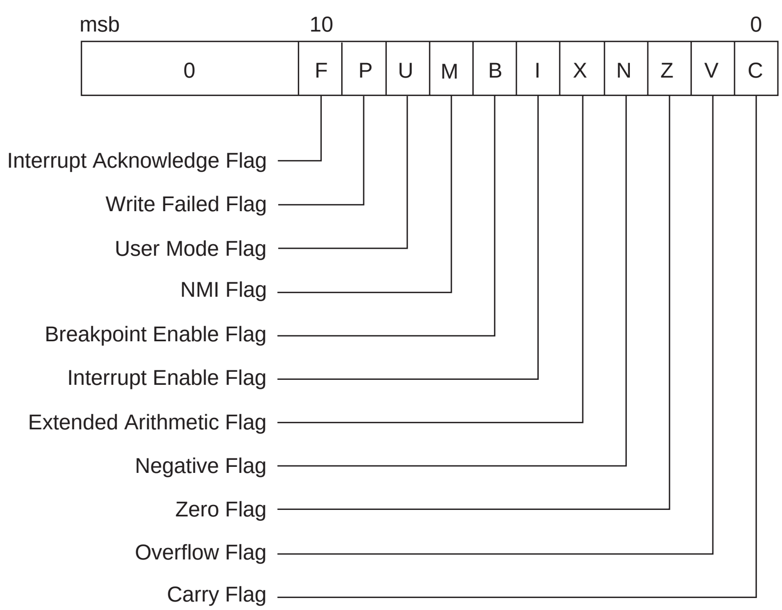


Figure 2-3 The Condition Code Register (CCR)/ Dword Condition Code Register (DCCR)

These flags can be tested using one of the 16 *condition codes* specified below:

Code	Alt	Condition	Encoding	Boolean Function
CC	HS	Carry Clear	0000	$\overline{C}$
CS	LO	Carry Set	0001	C
NE		Not Equal	0010	$\overline{Z}$
EQ		Equal	0011	Z
VC		Overflow Clear	0100	$\overline{V}$
VS		Overflow Set	0101	V
PL		Plus	0110	$\overline{N}$
MI		Minus	0111	N
LS		Low or Same	1000	C + Z
HI		High	1001	$\overline{C} * \overline{Z}$
GE		Greater or Equal	1010	$N * V + \overline{N} * \overline{V}$
LT		Less Than	1011	$N * \overline{V} + \overline{N} * V$
GT		Greater Than	1100	$N * V * \overline{Z} + \overline{N} * \overline{V} * \overline{Z}$
LE		Less or Equal	1101	$Z + N * \overline{V} + \overline{N} * V$
A		Always True	1110	1
WF		Write Failed	1111	P

Table 2-1 Condition Codes

2.3 Data Organization in Memory

The data types supported by the CRIS are:

Name	Description	Size Modifier
Byte	8-bit integer	.B
Word	16-bit integer	.W
Dword	32-bit integer or address	.D

Table 2-2 Data Types supported by the CRIS

Each address location contains one byte of data. Data is stored in memory with the least significant byte at the lowest address (“little endian”). The CRIS CPU in ETRAX 100LX has a 32-bit wide data bus. A conversion from 32 bits to 16 bits is performed by the bus interface in the case of an external 16-bit data bus mode.

Data can be aligned to any address. If the data crosses a 32-bit boundary, the CPU will split the data access into two separate accesses. So, the use of unaligned word and dword data will degrade performance.

The figures below show examples of data organization with a 16-bit bus and a 32-bit bus:

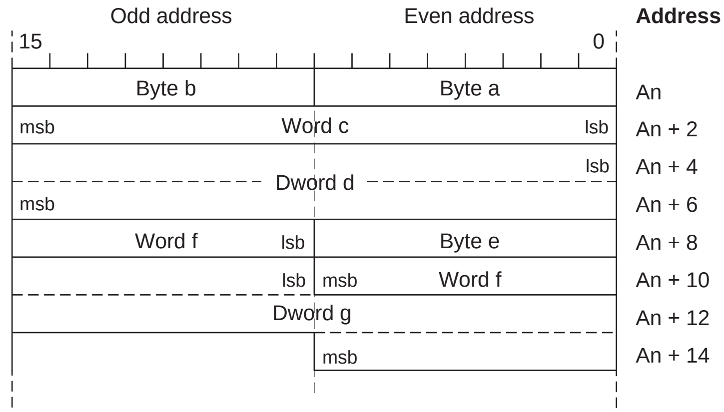


Figure 2-4 Example of Data Organization with a 16-bit Bus

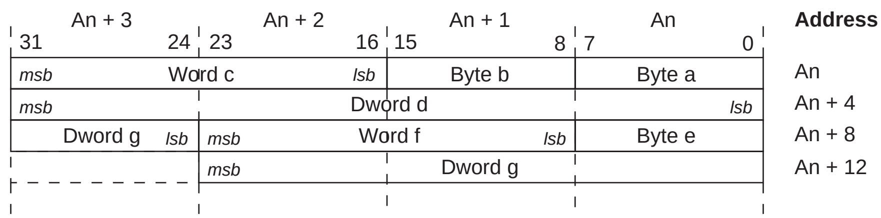


Figure 2-5 Example of Data Organization with a 32-bit Bus

2.4 Instruction Format

The basic instruction word is 16 bits long. Instructions must be 16-bit aligned.

When the CPU fetches 32 bits, containing two 16-bit aligned instructions, it saves the upper two bytes in an internal prefetch register. Thus, the CPU will only perform one read for every second instruction when running consecutive code.

The most common instructions follow the same general instruction format:

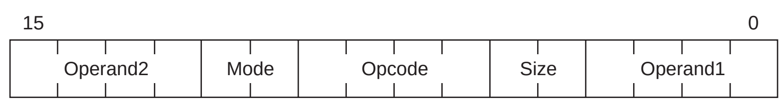


Figure 2-6 General Instruction Format

The following definitions apply to the instruction descriptions:

Syntax	Definition
m	Size modifier, byte, word or dword
z	Size modifier, byte or word
Rm	General register
Rn	General register
Rp	General register
Rs	Source operand, register addressing mode
[Rs]	Source operand, indirect addressing mode
[Rs+]	Source operand, auto increment addressing mode
s	Source operand, any addressing mode except quick immediate
si	Source operand, any mode except register or quick immediate
se	Source operand, indexed, offset, double indirect or absolute addressing mode
Pn	Special register
Ps	Source operand, special register
i	6-bit signed immediate operand
j	6-bit unsigned immediate operand
c	5-bit immediate shift value
Rd	Destination operand, register addressing mode
[Rd]	Destination operand, indirect addressing mode
[Rd+]	Destination operand, auto increment addressing mode
d	Destination operand, any addressing mode except quick immediate
di	Destination operand, any mode except register or quick immediate
Pd	Destination operand, special register
o	8-bit branch offset, bit 0 is the sign bit
x	8-bit signed immediate value
xx	16-bit signed immediate value
xxxx	32-bit signed immediate value
u	8-bit unsigned immediate value
uu	16-bit unsigned immediate value
uuuu	32-bit unsigned immediate value
cc	Condition code
n	4-bit breakpoint entry number

Table 2-3 Syntax Definitions

For a description of how the flags are affected, the following definitions apply:

Syntax	Definition
-	Flag not affected
0	Flag cleared
1	Flag set
*	Flag affected according to the result of the operation

Table 2-4 Flags Behavior Definitions

Instructions that do not have size modifiers operate on 32-bit data. The exception to this rule are those instructions that operate on 8-bit and 16-bit special registers.

2.4.1 Addressing Modes

The CRIS CPU has four basic addressing modes, which are encoded in the mode field of the instruction word. The basic addressing modes are:

- Quick immediate mode
- Register mode
- Indirect mode
- Autoincrement mode (with Immediate mode as a special case)

More complex addressing modes can be achieved by combining the basic instruction word with an addressing mode prefix word. The complex addressing modes are:

- Indexed
- Indexed with assign
- Offset
- Offset with assign
- Double indirect
- Absolute

The addressing modes of the CRIS CPU are described in the table below:

Assembler Syntax	Addressing Mode
i, j	Quick immediate
Rn	Register
Pn	Special register
[Rn]	Indirect
[Rn+]	Post increment
x, u	Byte immediate
xx, uu	Word immediate
xxxx, uuuu	Dword immediate
[Rn+Rm.m]	Indexed
[Rp=Rn+Rm.m]	Indexed with assign
[Rn+[Rm].m]	Indirect offset
[Rn+[Rm+].m]	Autoincrement offset
[Rn+x]	Immediate byte offset
[Rn+xx]	Immediate word offset
[Rn+xxxx]	Immediate dword offset
[Rp=Rn+[Rm].m]	Indirect offset with assign
[Rp=Rn+[Rm+].m]	Autoincrement offset with assign
[Rp=Rn+x]	Immediate byte offset with assign
[Rp=Rn+xx]	Immediate word offset with assign
[Rp=Rn+xxxx]	Immediate dword offset with assign
[[Rn]]	Double indirect
[[Rn+]]	Double indirect with auto increment
[uuuu]	Absolute

Table 2-5 The CRIS CPU Addressing Modes

2.4.2 Data Transfers

The data transfer instructions for the CRIS CPU, the two predefined assembler macros POP and PUSH, and the word/byte/bit SWAP instruction set are specified in table 2-6 below.

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
CLEAR.m	d	-	-	-	-	-	-	0	-	-	-	-	Clear destination operand
MOVE.m	s,Rd	-	-	-	-	-	-	0	*	*	0	0	Move from source to general register
MOVE.m	Rs,di	-	-	-	-	-	-	0	-	-	-	-	Move from general register to memory
MOVE (Pd == CCR/ DCCR)	s,Pd	*	*	*	-	*	*	0	*	*	*	*	Move from source to special register
MOVE (Pd!= CCR/ DCCR)	s,Pd	-	-	-	-	-	-	0	-	-	-	-	Move from source to special register
MOVE	Ps,d	-	-	-	-	-	-	0	-	-	-	-	Move from special register to destination
MOVEM	Rs,di	-	-	-	-	-	-	0	-	-	-	-	Move multiple registers to memory
MOVEM	si,Rd	-	-	-	-	-	-	0	-	-	-	-	Move from memory to multiple registers
MOVEQ	i,Rd	-	-	-	-	-	-	0	*	*	0	0	Move 6-bit signed immediate
MOVS.z	s,Rd	-	-	-	-	-	-	0	*	*	0	0	Move with sign extend
MOVU.z	s,Rd	-	-	-	-	-	-	0	0	*	0	0	Move with zero extend
POP	Rd	-	-	-	-	-	-	0	*	*	0	0	Pop register from stack
POP (Pd == CCR/ DCCR)	Pd	*	*	*	-	*	*	0	*	*	*	*	Pop special register from stack
POP (Pd!= CCR/ DCCR)	Pd	-	-	-	-	-	-	0	-	-	-	-	Pop special register from stack
PUSH	Rs	-	-	-	-	-	-	0	-	-	-	-	Push register onto stack
PUSH	Ps	-	-	-	-	-	-	0	-	-	-	-	Push special register onto stack
SBFS	di	-	-	-	-	-	-	0	-	-	-	-	Save bus fault status
SWAP <opt.>	Rd	-	-	-	-	-	-	0	*	*	0	0	Swap operand bits

Table 2-6 Data Transfer Instructions

Option	Description
N	Invert each bit
W	Swap the words of the operand
B	Swap the two bytes within each word of the operand
R	Reverse the bit order within each byte of the operand

Table 2-7 Options for the Word/Byte/Bit Swap Instruction

2.4.3 Arithmetic Instructions

The arithmetic instructions for the CRIS CPU are described in the table below:

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
ABS	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Absolute value
ADD.m	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Add source to destination register
ADDI	Rs.m,Rd	-	-	-	-	-	-	0	-	-	-	-	Add scaled index to base
ADDQ	j,Rs	-	-	-	-	-	-	0	*	*	*	*	Add 6-bit unsigned immediate
ADDS.z	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Add sign extended source to register
ADDU.z	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Add zero extended source to register
BOUND.m	s,Rd	-	-	-	-	-	-	0	*	*	0	0	Adjust table index (unsigned min)
CMP.m	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Compare source to register
CMPQ	i,Rd	-	-	-	-	-	-	0	*	*	*	*	Compare with 6-bit signed immediate
CMPS.z	si,Rd	-	-	-	-	-	-	0	*	*	*	*	Compare with sign extended source
CMPU.z	si,Rd	-	-	-	-	-	-	0	*	*	*	*	Compare with zero extended source
DSTEP	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Divide step
MSTEP	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Multiply step
MULS.m	Rs,Rd	-	-	-	-	-	-	0	*	*	*	0	Signed multiply
MULU.m	Rs,Rd	-	-	-	-	-	-	0	*	*	*	0	Unsigned multiply
NEG.m	Rs,Rd	-	-	-	-	-	-	0	*	*	*	*	Negate (2's complement)
SUB.m	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Subtract source from register
SUBQ	j,Rd	-	-	-	-	-	-	0	*	*	*	*	Subtract 6-bit unsigned immediate
SUBS.z	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Subtract with sign extended source
SUBU.z	s,Rd	-	-	-	-	-	-	0	*	*	*	*	Subtract with zero extended source
TEST.m	s	-	-	-	-	-	-	0	*	*	0	0	Compare operand with 0

Table 2-8 Arithmetic Instructions

2.4.4 Logical Instructions

The logical instructions for the CRIS CPU are described in the table below:

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
AND.m	s,Rd	-	-	-	-	-	-	0	*	*	0	0	Bitwise logical AND
ANDQ	i,Rd	-	-	-	-	-	-	0	*	*	0	0	AND with 6-bit signed immediate
NOT	Rd	-	-	-	-	-	-	0	*	*	0	0	Logical NOT (1's complement)
OR.m	s,Rd	-	-	-	-	-	-	0	*	*	0	0	Bitwise logical OR
ORQ	i,Rd	-	-	-	-	-	-	0	*	*	0	0	OR with 6-bit signed immediate
XOR	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Bitwise Exclusive OR

Table 2-9 Logical Instructions

2.4.5 Shift Instructions

The shift instructions of the CRIS CPU are shown in the table below. When the shift count is contained in a register, the 6 least significant bits of the register are used as an unsigned shift count.

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
ASR.m	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Right shift Rd with sign fill
ASRQ	c,Rd	-	-	-	-	-	-	0	*	*	0	0	Right shift Rd with sign fill
LSL.m	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Left shift Rd with zero fill
LSLQ	c,Rd	-	-	-	-	-	-	0	*	*	0	0	Left shift Rd with zero fill
LSR.m	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Right shift Rd with zero fill
LSRQ	c,Rd	-	-	-	-	-	-	0	*	*	0	0	Right shift Rd with zero fill

Table 2-10 Shift Instructions

2.4.6 Bit Test Instructions

The bit test instructions of the CRIS CPU are shown in the table below.

The BTST and BTSTQ instructions set the Z flag if the selected bit and all bits to the right of it are zero, and the N flag according to the selected bit in the destination register.

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
BTST	Rs,Rd	-	-	-	-	-	-	0	*	*	0	0	Test bit Rs in register Rd
BTSTQ	c,Rd	-	-	-	-	-	-	0	*	*	0	0	Test bit c in register Rd
LZ	Rs,Rd	-	-	-	-	-	-	0	0	*	0	0	Number of leading zeroes

Table 2-11 Bit Test Instructions

2.4.7 Condition Code Manipulation Instructions

The condition code manipulation instructions of the CRIS CPU are shown in the table below. The predefined assembler macros EI, DI and AX are also shown.

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
AX		-	-	-	-	-	-	1	-	-	-	-	Arithmetic extend (SETF X)
CLEARF	<list>	0	0	-	-	*	*	0	*	*	*	*	Clear flags in list
DI		0	0	-	-	-	0	0	-	-	-	-	Disable interrupts (CLEARF I)
EI		-	-	-	-	-	1	0	-	-	-	-	Enable interrupts (SETF I)
Scc	Rd	-	-	-	-	-	-	0	-	-	-	-	Set register according to cc
SETF	<list>	-	-	-	*	*	*	*	*	*	*	*	Set flags in list

Table 2-12 Condition Code Manipulation Instructions

2.4.8 Jump and Branch Instructions

The jump and branch instructions of the CRIS CPU and the predefined assembler macros RET, RETB and RETI are shown in the table below:

Instruction		Flag Operation											Description
		F	P	U	M	B	I	X	N	Z	V	C	
Bcc	o	-	-	-	-	-	-	0	-	-	-	-	Conditional relative branch
Bcc	xx	-	-	-	-	-	-	0	-	-	-	-	Branch with 16-bit offset
BREAK	n	1	-	*	-	-	-	0	-	-	-	-	Breakpoint
JBRC	s	-	-	-	-	-	-	0	-	-	-	-	Jump to breakpoint routine (note)
JIR	s	-	-	-	-	-	-	0	-	-	-	-	Jump to interrupt routine
JIRC	s	-	-	-	-	-	-	0	-	-	-	-	Jump to interrupt routine (note)
JMPU	si	-	-	-	-	-	-	0	-	-	-	-	Jump and set operation mode
JSR	s	-	-	-	-	-	-	0	-	-	-	-	Jump to subroutine
JSRC	s	-	-	-	-	-	-	0	-	-	-	-	Jump to subroutine (note)
JUMP	s	-	-	-	-	-	-	0	-	-	-	-	Jump
RBF	si	-	-	*	-	-	-	*	-	-	-	-	Return from bus fault
RET		-	-	-	-	-	-	0	-	-	-	-	Return from subroutine
RETB		-	-	-	-	-	-	0	-	-	-	-	Return from breakpoint routine
RETI		-	-	-	-	-	-	0	-	-	-	-	Return from interrupt routine

Table 2-13 Jump and Branch Instructions

Note: The JBRC, JIRC and JSRC instructions will add four bytes to the return address stored to either SRP, IRP or BRP. This leaves four bytes unused between the JSRC/JIRC/JBRC instruction and the return point. This can be used to enhance C++ exception support.

2.4.9 No Operation Instruction

Finally, the CRIS CPU also has a no operation instruction, NOP.

Instruction	Flag Operation											Description
	F	P	U	M	B	I	X	N	Z	V	C	
NOP	-	-	-	-	-	-	0	-	-	-	-	No operation

Table 2-14 No Operation Instruction

2.5 MMU Support

2.5.1 Overview

To support the Memory Management Unit (MMU) incorporated into the ETRAX 100LX, a number of features have been included in the CRIS architecture:

- The CPU can be in one of two different operation modes: *User mode* and *Supervisor mode*. The MMU uses the operation mode to select the appropriate mapping between logical and physical addresses.

- The *Bus fault* is a mechanism that can interrupt the CPU in any cycle, not only at instruction boundaries. This mechanism is needed because the MMU can get a page miss in any cycle.

- With the introduction of the bus fault mechanism, integral read-write operations can not be achieved by just disabling the interrupt. Instead, another method is used, see section 2.6 *Integral Read-Write Operations*.

The user and supervisor modes have different stack pointers. In both modes, the User mode Stack Pointer can be referenced as USP, while the currently active Stack Pointer is referenced as SP (or R14).

The following CRIS instructions are included specifically for MMU support:

- SBFS (Save Bus Fault Status)

- RBF (Return from Bus Fault)

- JMPU (Jump, set user mode if U flag is set)

The SBFS and RBF instructions are used at the entry and exit of the bus fault interrupt routine. They save and restore a 16 byte CPU status record containing the information necessary to resume the operation that was interrupted by the bus fault.

2.5.2 Protected Registers and Flags

A few registers and flags need to be protected from being modified while the CPU is in user mode. The protected registers and flags are:

- IBR (Interrupt Base Register)

- BAR (Breakpoint Address Register)

- M flag (NMI enable flag)

- B flag (HW breakpoint enable flag)

- I flag (Interrupt enable flag)

An attempt to modify a protected register while in user mode will just be silently denied. It will not cause any exception. The protected registers are readable in both user and supervisor modes.

2.5.3 Transition Between Operation Modes

A transition between the user and supervisor modes can take place for the following reasons:

Transition to user mode:

- JMPU with the U flag set
- RBF with the U flag set
- RETI with the U flag set
- RETB with the U flag set

Transition to supervisor mode:

- System Reset
- BREAK instruction
- Interrupt (including NMI and HW break)
- Bus fault

The stack pointers will be automatically exchanged at a transition between the user and supervisor modes.

2.5.4 Bus Fault Sequence

When the MMU signals a Bus Fault, the CPU will interrupt immediately at the end of the CPU clock cycle and enter a Bus Fault sequence. The Bus Fault sequence is similar to the ordinary interrupt sequence, see section 2.7 *Interrupts*.

The steps in the sequence are:

1. Bus Fault INTA cycle. This cycle will be an idle bus cycle.
2. Interrupt vector read cycle. The vector number will be 0x2e in MMU bus faults, and 0x20 in bus faults in the single step unit.
3. Start execution of the Bus Fault interrupt routine at the address given by the interrupt vector.

When entering into the Bus Fault interrupt routine, the internal CPU status is present in hidden CPU status registers. This status has to be saved to the memory using the SBFS instruction as the first instruction in the interrupt routine.

2.5.5 Format of the CPU Status Record

The format of the CPU status record is as follows:

31	0	
PC		An
Instruction	Execution State	An + 4
Interrupted Address		An + 8
Data		An + 12

Figure 2-7 CPU Status Record Format

- The *PC field* contains the value of PC immediately after the interrupted cycle. For example, if the bus fault occurs on an instruction fetch at address A in a linear instruction stream, the PC field will contain the value A + 2.

- The *Execution state field* contains a number of flags that enables the CPU to restart in the correct execution state. For more detail regarding these flags see Chapter 1 of the ETRAX 100LX Programmer's Manual.

- If the interrupted cycle was a data read or write (i.e. not an instruction fetch), the *Instruction field* contains the opcode of the interrupted instruction. If the interrupted cycle was an instruction fetch, the instruction field will contain the invalid data that was fetched during the interrupted cycle.

- The *Interrupted Address field* contains the address of the data entity in transfer during the interrupted cycle.

- Finally, the *Data field* contains data associated with the interrupted instruction. For more information see Chapter 1 of the ETRAX 100LX Programmer's Manual.

2.6 Integral Read-Write Operations

Since a bus fault can interrupt the CPU in any bus cycle (except INTA), it is not possible to ensure the integrity of a piece of code just by disabling the interrupts or by only using instructions that lock out interrupts between them. Instead, integral read-write operations can be implemented by using the *Load-Locked, Store-Conditional* principle discussed in the ETRAX 100LX Programmer's Manual.

Support for integral read-write operations includes the F and P flags, and a conditional write mechanism:

- The F flag (Interrupt acknowledge flag) is set by interrupts, bus faults, and the BREAK instruction.
- All instructions that write to memory, except for SBFS, can be made conditional. If both the F and X flags are set, no write will be performed and the P flag will be set instead.

2.7 Interrupts

The CRIS CPU uses vectorized interrupts that are generated either externally to, or internally by, the ETRAX 100LX. The interrupt acknowledge sequence is as follows:

1 Perform an INTA cycle, where the 8-bit vector number is read from the bus.

2 Store the contents of PC to the Interrupt Return Pointer (IRP). Note that the return address is not automatically pushed on the stack.

3 Read the interrupt vector from the address [IBR + <vector number> * 4].

4 Start the execution at the address pointed to by the interrupt vector.

The Interrupt Base Register (IBR) has bits 31-16 implemented. The remaining bits are always zero.

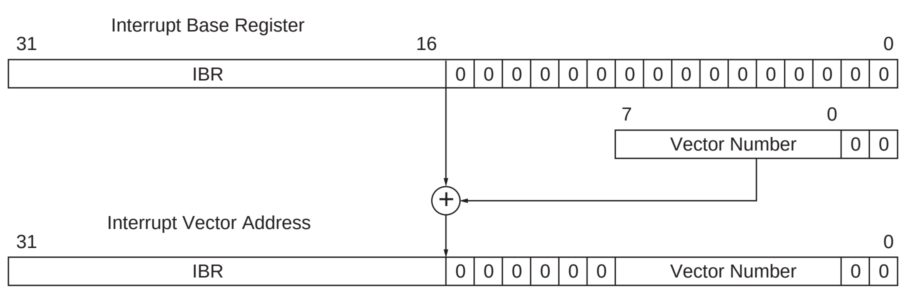


Figure 2-8 Interrupt Vector Address Calculation

2.7.1 NMI

The Non Maskable Interrupt (NMI) is handled in the same way as the normal interrupt except for the following three differences:

- The return address is stored in the Breakpoint Return Pointer (BRP) instead of the IRP.
- The NMI is enabled/disabled by the M flag instead of the I flag. The M flag can be set with the SETF M instruction. Move to CCR/DCCR has no effect. Once set, the M flag can only be cleared by an NMI acknowledge cycle or system reset.
- The INTA cycle will be an idle bus cycle, and the vector number 0x21 is generated internally in the CPU.

2.8 Software Breakpoints

The CRIS CPU has a breakpoint instruction (BREAK n). This instruction saves the current value of PC in the Breakpoint Return Pointer (BRP) register and performs a jump to address (IBR + 8 * n).

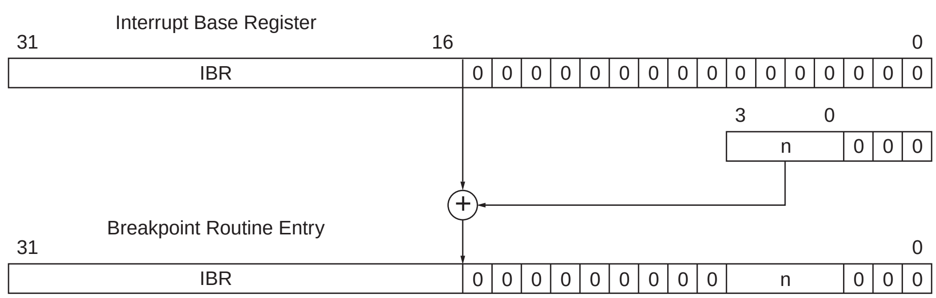


Figure 2-9 Breakpoint Routine Entry Address Calculation

2.9 Hardware Breakpoint Mechanism

The CPU contains a hardware breakpoint mechanism. The hardware breakpoint address is loaded in the BAR register (Breakpoint Address Register), and the hardware breakpoint mechanism is enabled by setting the Breakpoint enable flag B (see Figure 2-2 on page 6).

For each CPU read or write cycle, the address is compared with the contents of the BAR register. In order to detect a read or write in the dword (and not just a single byte) of the address location, bit 1 and 0 are ignored in the comparison. Bit 31 is also ignored in the comparison since that bit handles the cache in the ETRAX 100LX (address bit 31 set will bypass the cache and directly access the main memory).

An address hit is handled in the same way as an NMI with interrupt vector number 0x20, except that a breakpoint hit is not affected by the M flag.

The hardware breakpoint mechanism is disabled after reset.

3 SINGLE STEP

3.1 General

There is a simple single step mechanism in the ETRAX 100LX. The single step is controlled by 4 mode bits (19:16) in the R_TEST_MODE register.

Bit nr.	Name	Description
19	single_step	Turns single step on and off.
18	break_write	Enables the single step unit to break execution on memory write cycles.
17	break_read	Enables the single step unit to break execution on memory data read cycles (as opposed to instruction fetch).
16	break_fetch	Enables the single step unit to break execution on instruction fetch cycles. This will break both fetch from memory and fetch from the CPU internal prefetch register.

Table 3-1 Single Step Control Bits in the R_TEST_MODE Register

The single step unit is started by first setting the appropriate bits in the R_TEST_MODE register, and then using the RBF instruction to jump to the code through which the single step should be performed.

The single step unit will issue a bus fault on the first CPU cycle that matches the cycle types selected in the R_TEST_MODE register.

The single step bus fault uses the interrupt vector number 0x20, which is the same interrupt vector that is used by the HW break mechanism. As a result, it is not recommended that the single step and the HW break mechanisms be enabled at the same time.

After the bus fault has occurred, the single step unit is disabled until after the next RBF instruction has been executed and the restarted cycle has been completed.

If a single step bus fault occurs at the same time as an MMU bus fault, the single step bus fault will have priority. This means that the single step routine must be able to check for this case and invoke MMU handling as well, but the MMU handler doesn't need to be aware of the single step mechanism. To check for concurrent MMU and single step bus faults, there is a both_faults bit (R_BUS_STATUS bit 4) that is set when this happens. The both_faults bit is cleared when the CPU runs an RBF instruction.

3.2 Programming Considerations

A special case occurs with interrupts when the break_read bit is set. The single step unit will then break also on interrupt vector read cycles. This case can be identified by a bit in the CPU status record, but the interrupt sequence cannot be automatically restarted by the RBF instruction. The single step handler routine must check for this case and compensate for it.

When single stepping through code that uses the integral read-write mechanism, the 'old F flag' in the CPU status record must be checked, and the F flag in CCR must be restored if there wasn't a concurrent MMU bus fault. Otherwise, the conditional write will always fail in single step mode, and the program will hang.

Another problem that has to be taken care of occurs when single stepping through code that pops data from the supervisor stack. After a bus fault in the data read cycle of a pop, the stack pointer will already be incremented. If the single step handler uses the stack this will clobber the stack data that was going to be read when the bus fault occurred.

There are two possible solutions for this problem:

1 Reserve extra space on the stack.

Example:

```
SBFS [SP = SP - 80]                ; Reserve 64 extra bytes
PUSH DCCR
DI
     :
     :
     :
MOVE [SP = SP + 84], DCCR          ; Dummy read to adjust SP, and
                                   ; lock out irq.
MOVE [SP - 84], DCCR               ; "POP" DCCR
RBF [SP - 80]
```

This will reserve stack space corresponding to the maximum SP increment that could occur with Autoincrement addressing mode. The maximum of 64 bytes occurs with the MOVEM [SP+], PC instruction.

This method will only work as long as the program you are single stepping through doesn't itself use the same method as described above to access values on the stack after the stack pointer has been incremented.

1 Store status at absolute address and have a separate stack for the single step handler.

Example:

```
SBFS [STATUS_RECORD]               ; Save CPU status to
                                   ; absolute address.
MOVE DCCR, [SAVED_DCCR]            ; Save DCCR to absolute address.
DI
MOVEM SP, [SAVED_REGS]             ; Save all registers to
                                   ; absolute address.
MOVE.D SINGLE_STEP_SP, SP          ; Load new sp value for
                                   ; single step handler.
:
:
:
MOVEM [SAVED_REGS], SP
MOVE [SAVED_DCCR], DCCR
RBF [STATUS_RECORD]
```

4 Memory Management Unit

The ETRAX 100LX includes a Memory Management Unit (MMU) that manages the physical memory resources of the device. The MMU has a number of purposes:

- Protecting the code and data of one user-level application from other user-level applications.
- Permitting user-level applications to share portions of the address space.
- Protecting the kernel-level code and data from user-level applications.
- Running applications that are partially resident in main memory. Only the most recent part of such an application is normally stored in main memory: the rest of the program is stored on disk until needed. This is more commonly known as swapping.
- Reference counting in support of a garbage collection mechanism.

The MMU implements a virtual memory system where it creates the illusion of a very large amount of memory exclusively available for each application.

In ETRAX 100LX, 4 GBytes of virtual memory are made available to each application. Each application is also given a separate address space identifier that is used by the MMU to separate its memory from other applications.

The virtual memory is divided into 8 KByte pages which can be individually protected and mapped to physical memory. The upper 19-bit part of the 32-bit CPU address is known as the virtual page number (vpn), while the lower 13-bit part is used as an offset within each page. The vpn is translated into a 19-bit physical page number (pfn) while the page offset remains unchanged through the MMU. The MMU uses the CPU’s *User* and *Supervisor Modes* to restrict access and select the appropriate mapping from logical to physical addresses in the virtual memory space.

Mapping from virtual address to physical address is handled by a *Translation Lookaside Buffer* (TLB). The TLB is an on-chip cache that provides translations in the form of page table entries (pte). The ptes are stored in page tables in main memory. If a virtual-to-physical address translation is not found in the TLB, the MMU will interrupt the CPU and use a software table walker to look for the translation in the main memory page table.

In some cases the MMU is not used, so it can be disabled in register R_MMU_ENABLE. By default the MMU is disabled after reset. When the MMU is disabled, the ETRAX 100LX is considered to be in *non-protected* mode and memory is accessed in the conventional manner.

Please see chapter 18.17 *MMU Registers* for detailed information on the MMU registers.

4.1 MMU Memory Areas

The MMU operates with two types of memory area:

- Kernel/user area - accessed in the CPU User and Supervisor Modes, and mainly contains user code and data structures.
- Kernel area - accessed in the CPU Supervisor Mode only and may include kernel code and data structures, I/O-buffers, DMA-buffers, mode registers, etc.

4.1.1 Kernel/User Address Space

The *kernel/user* address space is a uniform, virtual address area of 4 GBytes in size. It is divided into 8 KByte pages that can be individually protected and mapped to physical memory. Mapping is executed through the TLB, which translates a virtual address into a corresponding physical address. Each user process has a unique translation of virtual addresses placed in page tables in main memory. The page tables are maintained by the kernel.

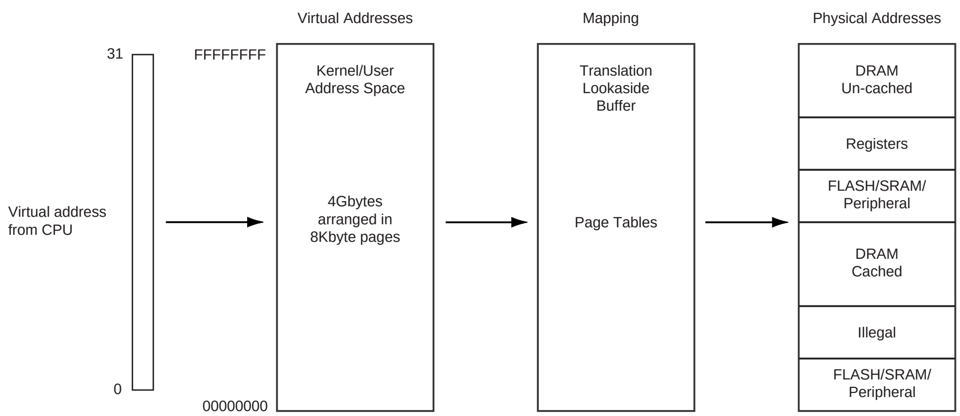


Figure 4-1 Kernel/User Virtual Address Space

4.1.2 Kernel Address Space

The kernel address space is divided into sixteen 256 MByte segments designated seg_0 to seg_f. Each segment can be individually configured to use page mapping or linear segment mapping. The mapping method is determined in kernel segment register R_MMU_KSEG. When a bit in this register is set to 0, the corresponding segment is page-mapped via the TLB. When a segment bit is set to 1, the corresponding segment is linear mapped by means of a 4-bit offset. Linear-mapped segments do not use the TLB.

Page-Mapped Kernel Segments

Kernel address segments that use page mapping are divided into 8 KByte pages controlled by the TLB. The four-bit base fields in registers R_MMU_KBASE_LO and R_MMU_KBASE_HI are ignored for page mapping purposes.

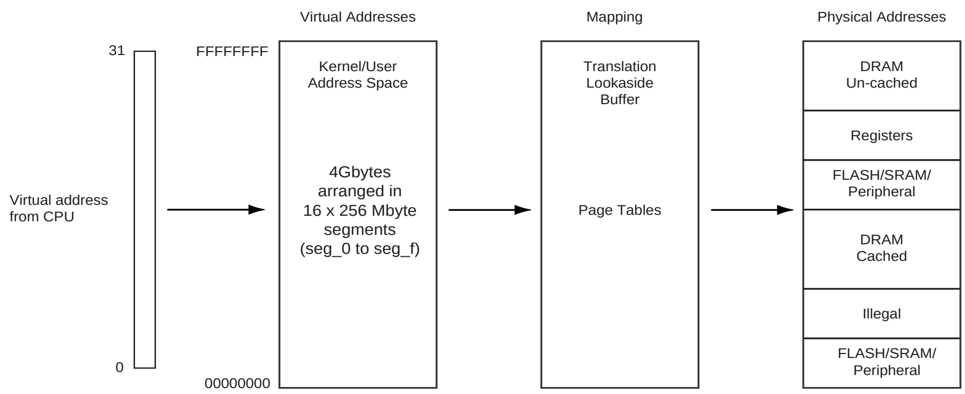


Figure 4-2 Kernel Page-mapped Virtual Address Space

Linear Segment-Mapped Kernel Areas

The virtual addresses of kernel segments that use linear mapping are converted to physical addresses by translating the four m.s.b. of the 32-bit virtual address from the CPU. The linear translation is handled in two registers, R_MMU_KBASE_HI and R_MMU_KBASE_LO. The 4-bit base field in these registers becomes address bits 31 to 28 when translating to physical addresses. Bits 27 to 0 of the virtual address are unchanged when translated to a physical address.

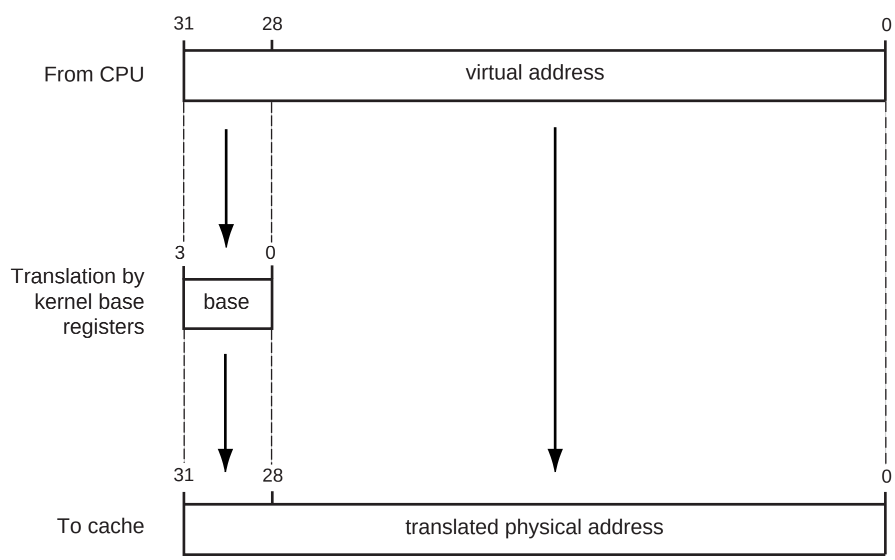


Figure 4-3 Linear Segment Address Translation

4.2 Translation Lookaside Buffer

The Translation Lookaside Buffer is a 64-entry cache that maps a virtual address to a physical address. If a translation cannot be found in the TLB (a *TLB miss*), the CPU is interrupted (chapter 2, refers) and the software is required to load a new translation into the TLB.

4.2.1 TLB Memory Sets

The TLB comprises four 16-entry memory sets designated 0 to 3. The TLB is thus four-set associative. This means that a vpn from the CPU can be stored at only one location in each of the four TLB memory sets, and in each set the chosen location is the same (see Figure 4.4).

An example of possible locations for different addresses in the TLB is given in the table below.

VPN	Positions in TLB			
0	0	16	32	48
1	1	17	33	49
2	2	18	34	50
. . . .	. . . .	. . . .	. . . .	. . . .
14	14	30	46	62
15	15	31	47	63
16	0	16	32	48
17	1	17	33	49

Table 4-1 Example of Virtual Address Positions in the TLB

The selection of the TLB set in which a particular vpn will be stored must be configured in software. The location at which the vpn will be placed within the TLB set is determined by a 4-bit index comprising bits 13 to 16 of the incoming virtual address from the CPU.

In the event of a TLB miss, a 2-bit set number is provided by a random number generator. The random set number is combined with the 4-bit index from the CPU address to form a 6-bit index. This index is stored in register R_TLB_SELECT, and can be used to choose which of the 64 TLB entries to replace. It is also possible for the software to use another algorithm to select the entry to replace. As shown below, bits 4 and 5 of the index provide the TLB set number and bits 3 to 0 denote the location within the set.

set		location			
5	4	3	2	1	0

Figure 4-4 Format of TLB Miss Exception Index

To accommodate the 8 KByte page size, the lower 13 bits of the 32-bit virtual address from the CPU are not changed by the translation. The 19-bit virtual page number is translated into a 19-bit physical frame number.

The principle of the TLB memory sets is illustrated below.

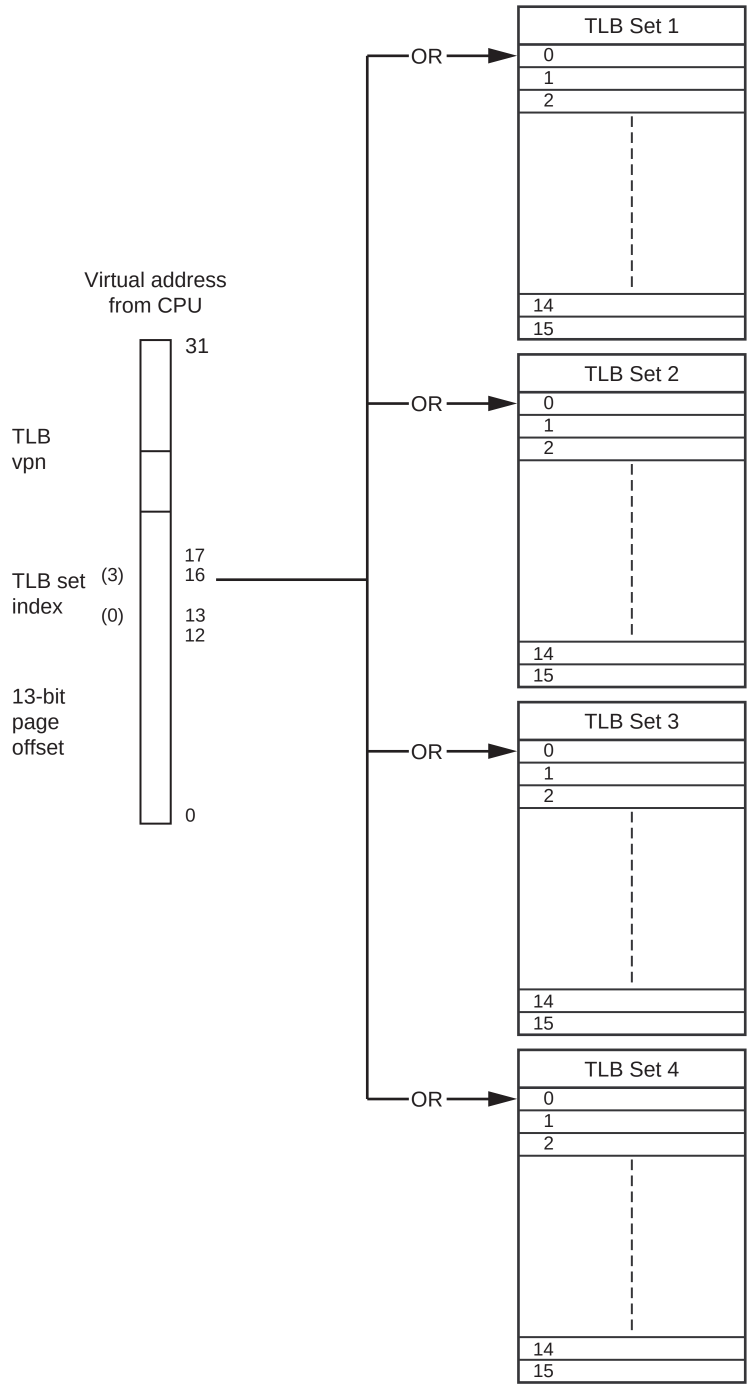


Figure 4-5 TLB Memory Sets

4.2.2 TLB Entries

Entries stored in the TLB are 44 bits in width. They consist of a virtual to physical address translation, a page identification, and control bits. The format of a TLB entry is shown below.

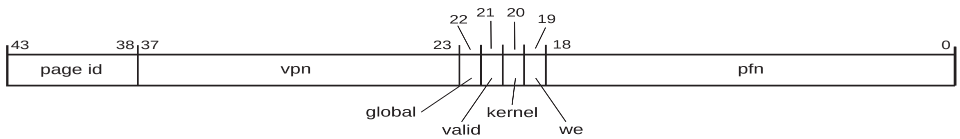


Figure 4-6 Format of a TLB Entry

- pfn - the translated address expressed as a 19-bit physical frame number;
- vpn - the virtual address expressed as a 15-bit virtual page number: the remaining 4 bits are composed from the 4-bit TLB set index.
- page_id - the 6-bit page identification;
- we - write enable bit 19 is used to write-protect the page;
- kernel - bit 20 is used to prevent access to the page during CPU User Mode;
- valid - bit 21 indicates that the TLB entry contains a valid address translation;
- global - bit 22 indicates whether or not the TLB ignores the page identification matching conditions for a TLB hit.

If the global bit is not asserted (0), the page_id in the TLB entry is compared to the value of the 6-bit page_id in register R_MMU_CONTEXT. To achieve a valid translation, the values of the page_id fields in the TLB entry and the register must match.

If the global bit is asserted (1), the page_id of the TLB entry is not compared to the page_id field in register R_MMU_CONTEXT.

The condition for a hit in one of the TLB sets is:

```
hit = (valid | inv_excp) &
      (cpu_vpn == tlb_vpn) &
      ((context_pid == tlb_pid) | global))
```

where:

valid	valid bit (21) in the TLB entry.
inv_excp	invalid page exception enabling bit from R_MMU_CTRL.
cpu_vpn	virtual page number from the CPU.
tlb_vpn	virtual page number in the TLB entry.
context_pid	page_id field in R_MMU_CONTEXT.
tlb_pid	page_id field in the TLB entry.
global	global bit (22) in the TLB entry.

Table 4-2 Definitions of TLB Hit

4.2.3 TLB Register Interface

The TLB is controlled by registers R_TLB_SELECT, R_TLB_LO and R_TLB_HI, all entries to which can be read and written by the CPU. Register R_TLB_SELECT is used to choose which entry is to be read or written in the TLB.

TLB entries are 44 bits in width and therefore an entry cannot be written by the CPU in one cycle. The CPU must first write the high order part of the TLB entry to register R_TLB_HI. This contains the same fields as register R_MMU_CAUSE and, when writing to R_TLB_HI, the data are also stored at the corresponding fields of R_MMU_CAUSE. The high order part is not stored in the TLB until the low order part is written in register R_TLB_LO.

Registers R_MMU_CAUSE and R_TLB_SELECT are normally updated automatically by the MMU and do not require updating by the software. To write a new translation in the TLB, for example a TLB miss, the software only has to write into register R_TLB_LO.

4.2.4 Virtual Address from the CPU

An incoming address from the CPU is constructed as shown below.

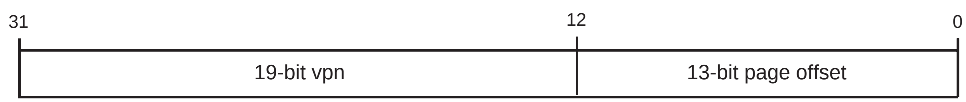


Figure 4-7 Incoming Address from the CPU

The page offset is 13 bits wide to accommodate the offset within the 8 KByte pages. The upper 15 bits of the vpn are stored in the TLB and the lower four bits are the address index for the selected TLB set.

When the CPU generates a new address, the four TLB sets are searched for a matching vpn. If an entry is found, the page_id is compared to the corresponding field in register R_MMU_CONTEXT. The valid, kernel and we bits are controlled and, if all conditions match, a valid physical address is output to the cache.

The TLB is not permitted to store more than one valid, virtual page translation with the same page_id or with the global bit set. This would cause multiple hits in the TLB and result in an MMU exception.

4.2.5 MMU Exceptions

An exception is generated if there is a mismatch in the output from one or more fields of the TLB. In the event of an exception, the MMU interrupts the CPU by means of bus fault logic (See chapter 2 *RISC CPU*). Information about the exception is stored in register R_MMU_CAUSE, which holds information about:

- The vpn that generated the exception
- The page_id of the application that generated the exception
- Whether the exception was caused by a write or read access

- The type of exception that occurred

Five different types of MMU exception can occur (Refer to section *4.2.1 TLB Memory Sets* for those exceptions that generate a random or actual index):

- Miss - The referenced address does not match any TLB entry, or the current page_id does not match the TLB index. A valid entry must be loaded by software.
- Write error - During a write operation, reference is made to a page that does not have the we bit asserted in the TLB entry. This exception can be used for write protection and dirty checks.
- Access violation - In User Mode, reference is made to a page with the kernel bit asserted in the TLB entry. When asserted, the kernel bit prevents CPU User Mode access to the page.
- Invalid page - Reference is made to a page with matching vpn and page_id fields in the TLB, but the valid bit is deasserted. This can be used for reference counting.
- Multiple hits - The occurrence of more than one hit in the TLB indicates a serious error. This is indicated by a bus fault signal with all exception bits deasserted.

The write error, access violation and invalid page exceptions can all be disabled in control register R_MMU_CTRL. Normally the invalid page exception is disabled unless a reference count is in progress. When disabled, an invalid entry in the TLB will not match any address and will, thus, cause a miss.

The nature of a miss exception is such that it does not allow any other exceptions to occur simultaneously. However, a single memory reference may cause any combination of write error, access violation and invalid page exceptions at the same time.

The conditions for the MMU exceptions are:

- Miss = ~hit
- Write error = hit & cpu_wr & ~we & we_excp
- Access violation = hit & user_mode & kernel & acc_excp
- Invalid page = hit & ~valid & inv_excp

where:

hit	Hit in one of the TLB sets.
cpu_wr	CPU write access.
we	Write enable bit in TLB entry.
we_excp	Write error exception enabling bit from R_MMU_CTRL.
user_mode	CPU User Mode access.
kernel	Kernel bit in TLB entry.
acc_excp	Access violation exception enabling bit from R_MMU_CTRL.
valid	Valid bit in TLB entry.
inv_excp	Invalid page exception enabling bit from R_MMU_CTRL.

Table 4-3 Definition of MMU Exceptions

4.3 MMU Registers

The MMU is served by a set of dedicated registers. The table below summarizes the purpose of each register.

Register	Function
R_MMU_CONFIG comprising:	
R_MMU_KSEG	Sets individual page or segment mapping method for each kernel segment 0 to f.
R_MMU_CTRL	Enables or disables the invalid page (valid), access (kernel) and write-enable (we) MMU exceptions.
R_MMU_ENABLE	Enables or disables the MMU.
R_MMU_KBASE_LO	Provides the 4-bit offset for linear translations in the eight lower kernel segments 0 to 7.
R_MMU_KBASE_HI	Provides the 4-bit offset for linear translations in the eight upper kernel segments 8 to f.
R_MMU_CONTEXT	Contains the 6-bit page_id of the current address map.
R_MMU_CAUSE	Multi-purpose, containing: the 19-bit vpn of the referenced address that generated an exception when an MMU exception occurs; the 6-bit page_id from R_MMU_CONTEXT when an MMU exception occurs; 4 discrete bits signifying whether a miss, invalid, access or write-enable exception has occurred; 1 bit signifying whether the exception was caused by a write or read access.
R_TLB_SELECT	Contains the 6-bit TLB index.
R_TLB_LO	Contains the lower 23 bits of the TLB entry, namely: 19-bit pfn; global bit; valid bit; kernel bit; we bit.
R_TLB_HI	Contains the upper 25 bits of the TLB entry, namely: 19-bit vpn; 6-bit page_id.

Table 4-4 MMU Registers

For detailed information on the MMU registers, please refer to chapter 18.17 *MMU Registers*.

4.4 MMU Test Mode

The MMU can be set to a test mode that can be used to manually check the TLB and the bus fault logic. The MMU test mode is set by bit mmu_test in register R_TEST_MODE.

When the MMU test mode is active, the bus fault logic is operative but the bus fault signal to the CPU is gated and cannot, therefore, generate an interrupt to the CPU. Register R_MMU_CAUSE is updated as normal.

The address translation logic is not affected by the MMU test mode. If the MMU is enabled, mapping through the TLB and offset registers operates in the normal way. If the MMU is disabled, the output physical address is not translated.

mmu_en	mmu_test	Interrupt CPU	Update R_MMU_CAUSE	Translate Address
0	0	no	no	no
0	1	no	yes	no
1	0	yes	yes	yes
1	1	no	yes	yes

Table 4-5 MMU Test Mode Truth Table

4.5 Example of Virtual Memory Configuration

Virtual memory configuration is essentially a matter of setting up the ratio of page-mapped kernel space to linear-mapped kernel space. For example, to set up the following virtual memory system, the MMU register configurations would resemble those described in the tables below.

- 0.5 GBytes of linear-mapped kernel area;
- 3.5 GBytes of page-mapped kernel area;
- 4 GBytes of kernel/user area.

Register R_MMU_KBASE_HI

Bit(s)	Name	Setting	Value
31 - 28	base_f	Don't care.	0 x 0
27 - 24	base_e	Don't care.	0 x 0
23 - 20	base_d	Don't care.	0 x 0
19 - 16	base_c	Kernel cached area.	0 x 7
15 - 12	base_b	Kernel uncached mode registers.	0 x b
11 - 8	base_a	Don't care.	0 x 0
7 - 4	base_9	Don't care.	0 x 0
3 - 0	base_8	Don't care.	0 x 0

Table 4-6 Example of Kernel Base High Register Configuration

Register R_MMU_KBASE_LO

Bit(s)	Name	Setting	Value
31 - 28	base_7	Don't care.	0 x 0
27 - 24	base_6	Don't care.	0 x 0
23 - 20	base_5	Don't care.	0 x 0
19 - 16	base_4	Don't care.	0 x 0
15 - 12	base_3	Don't care.	0 x 0
11 - 8	base_2	Don't care.	0 x 0
7 - 4	base_1	Don't care.	0 x 0
3 - 0	base_0	Don't care.	0 x 0

Table 4-7 Example of Kernel Base Low Register Configuration

Register R_MMU_KSEG

Bit(s)	Name	Setting	Value
15	seg_f	Page mapping.	0
14	seg_e	Page mapping.	0
13	seg_d	Page mapping.	0
12	seg_c	Kernel linear segment mapping.	1
11	seg_b	Kernel linear segment mapping.	1
9	seg_a	Page mapping.	0
8	seg_9	Page mapping.	0
7	seg_7	Page mapping.	0
6	seg_6	Page mapping.	0
5	seg_5	Page mapping.	0
4	seg_4	Page mapping.	0
3	seg_3	Page mapping.	0
2	seg_2	Page mapping.	0
1	seg_1	Page mapping.	0
0	seg_0	Page mapping.	0

Table 4-8 Example of Kernel Segment Register Configuration

Memory maps of the virtual address spaces realized by the example configuration are illustrated below.

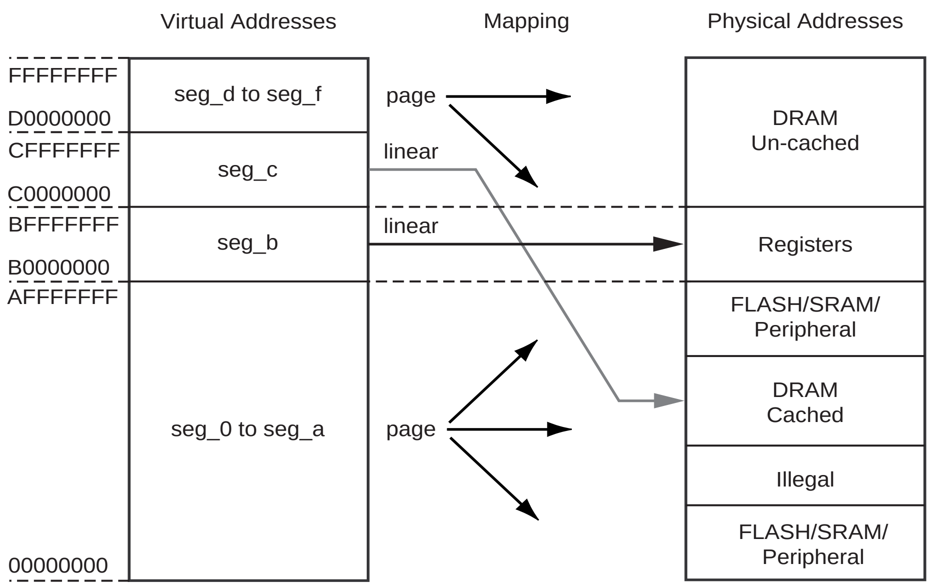


Figure 4-8 Example of Kernel Virtual Memory Map

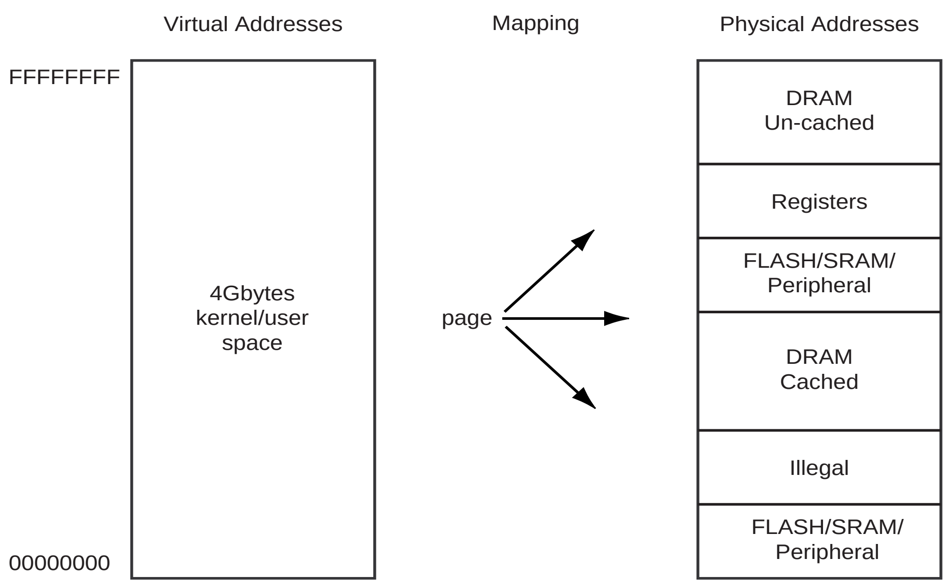


Figure 4-9 Example of Kernel/User Virtual Memory Map

5 BUS INTERFACE

The ETRAX 100LX bus interface has a 32/16-bit data bus, a 25-bit address bus, and six internally decoded chip select outputs. Six additional chip select outputs are multiplexed with other I/O functions, and are available in some configurations. The bus interface also supports either asynchronous or synchronous DRAM banks without external logic.

5.1 Data Bus

The 32-bit data bus provides support for 16-bit wide memories. The data bus is organized with the least significant byte at the lowest address ("little endian").

The 32-bit data bus mode is shown in figure 5-1, and the 16-bit data bus mode is shown in figure 5-2 below:

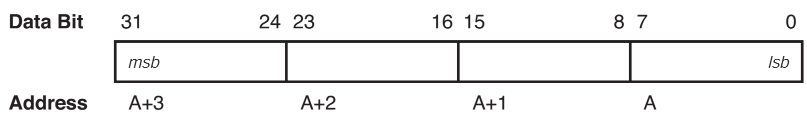


Figure 5-1 32-bit Mode Data Bus

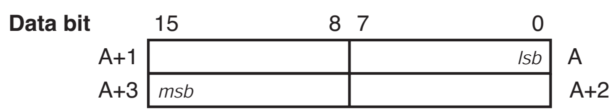


Figure 5-2 32-bit Wide Data on a 16-bit Mode Data Bus

5.2 Bus Interface Registers

Register	Function
R_WAITSTATES	A 32-bit write only register containing waitstate settings for peripheral, SRAM, and flash-PROM chip selects.
R_BUS_CONFIG	A 32-bit write only register for selecting bus width, common write enable (cwe) or bytewise write enable (bwe) selection, and DMA burst length,. It is also used for setting write delay mode for chip selects.
R_BUS_STATUS	A 32-bit read only register that shows the initial values of the bus status pins bs0 - bs3 after reset.
R_DRAM_TIMING	A 32-bit write only register for asynchronous DRAM waitstate configuration.
R_SDRAM_TIMING	A 32-bit write only register for SDRAM enabling and configuration.
R_DRAM_CONFIG	A 32-bit write only register for asynchronous DRAM bus width, and DRAM type and bank selection.
R_SDRAM_CONFIG	A 32-bit write only register for SDRAM bus width, and SDRAM type and bank selection.

Table 5-1 Bus Interface Registers

For more detailed information see chapter 18.2 *Bus Interface Configuration Registers*.

5.3 Address and Chip Selects

The ETRAX 100LX chip operates with a 32-bit wide address space internally, but only address bits 25-1 are available on the external pins. Address bits 30-26 are decoded internally to generate the different memory chip select outputs, and the select of DRAM banks and internal I/O. Address bit 31 is used to select whether the cache is used or bypassed by CPU accesses. Additional chip selects are configured in the register R_PORT_PB_SET (See chapter 18.2 *Bus Interface Configuration Registers*).

The addresses are decoded to generate chip selects as follow:

Address Range (hex)	Size (Mbyte)	Name	Description
00000000-03FFFFFF	64	cse0	EPROM/flash PROM bank 0 chip select (Note: 1)
04000000-07FFFFFF	64	cse1	EPROM/flash PROM bank 1 chip select (Note: 1)
08000000-0BFFFFFF	64	csr0	SRAM bank 0 chip select. (Note: 1)
0C000000-0FFFFFFF	64	csr1	SRAM bank 1 chip select (Note: 1)
10000000-13FFFFFF	64	csp0	Peripheral chip select 0 (Note: 1)
14000000-17FFFFFF	64	csp1	Peripheral chip select 1 (Note: 1) Note: 2)
18000000-1BFFFFFF	64	csp2	Peripheral chip select 2 (Note: 1) (Note: 2)
1C000000-1FFFFFFF	64	csp3	Peripheral chip select 3 (Note: 1) (Note: 2)
20000000-23FFFFFF	64	csp4	Peripheral chip select 4 (Note: 2)
24000000-27FFFFFF	64	csp5	Peripheral chip select 5 (Note: 1) (Note: 2)
28000000-2BFFFFFF	64	csp6	Peripheral chip select 6 (Note: 1) (Note: 2)
2C000000-2FFFFFFF	64	csp7	Peripheral chip select 7 (Note: 1) (Note: 2)
30000000-3FFFFFFF	256	-	Do not use (Note: 3)
40000000-7FFFFFFF	1024	-	DRAM interface (Note: 1)
80000000-AFFFFFFF	768		Same as 00000000-2FFFFFFF but uncached
B0000000-B7FFFFFF	128	-	Internal registers
B8000000-BFFFFFFF	128	-	Internal start up code
C0000000-FFFFFFFF	1024	-	Same as 40000000-7FFFFFFF but uncached

Table 5-2 Chip Selects

Note 1: Add 80000000 (hex) to bypass the cache.

Note 2: Peripheral select 1 - 3 and 5 - 7 are multiplexed with bits 2 - 7 in general port PB in the I/O block, and are not available if these pins are configured as general port pins.

Note 3: This region is internal registers and start-up code, but it is cached. Never use this area for accessing internal registers.

5.4 Internal Bus Arbitration

The bus interface performs the bus arbitration between the possible internal bus masters. The bus priority order is:

1 Shared RAM interface *(highest priority)*
2 External DMA channels
3 DRAM refresh
4 Internal DMA channels
5 CPU and cache *(lowest priority)*

5.5 Bus Width, Cycle Timing and Wait States

The bus interface supports 4 asynchronous DRAM banks, 8 synchronous DRAM banks (i.e. 2 groups of SDRAM banks with 2 to 4 banks per group), and between 6 to 12 other memory banks depending on its configuration. Each memory bank is connected to one of the chip select outputs. The banks are separated into five groups:

Group 1	$\overline{cse0}$, $\overline{cse1}$
Group 2	$\overline{csr0}$, $\overline{csr1}$
Group 3	$\overline{csp0}$, $\overline{csp1}$, $\overline{csp2}$, $\overline{csp3}$
Group 4	$\overline{csp4}$, $\overline{csp5}$, $\overline{csp6}$, $\overline{csp7}$
Group 5	DRAM banks

Table 5-3

The ETRAX 100LX bus width and number of wait states can be configured individually for each group. The bus width can be configured to either 16 or 32 bits using the internal register R_BUS_CONFIG. All banks in the same group have the same width. For group 1, the EPROM/flash PROM group, the initial bus width is decided by bus status pin 0 (bs0) at system reset. If bs0 is low, the bus width is configured to 16 bits. If bs0 is high, then the bus width is configured to 32 bits.

All bus cycles are run in consecutive bursts, with a maximum length of 32 bytes, and a minimum length of 1 byte. A burst will never cross a 32-byte boundary. A burst is either read or write, and the data direction is never changed within a burst.

For each group of memory banks except Synchronous DRAM banks, the following wait state parameters can be configured:

Parameter	Name	Size (bits)	Description
Early wait states	ew	2	Number of wait states before the falling edge of $\overline{rd}$, $\overline{wr0}$ - $\overline{wr3}$, or $\overline{cas}$.
Late wait states	lw	2 or 4 (Note: 44)	Number of wait states after the falling edge of $\overline{rd}$, $\overline{wr0}$ - $\overline{wr3}$, or $\overline{cas}$.
Turn-off wait states	zw	2	Number of wait states after the end of a burst. The turn-off wait states of the ending burst and the early wait states of the next burst may overlap.

Table 5-4 Memory Bank Wait State Parameters That Can Be Configured

Note 4: There are 2 bits for DRAM, and 4 bits for all other memory banks.

Group or Cycle	Wait State Ranges		
	Early	Late	Turn Off
Group 1: $\overline{\text{cse0}}$, $\overline{\text{cse1}}$	0 – 3	0 – 15	0 – 3
Group 2: $\overline{\text{csr0}}$, $\overline{\text{csr1}}$	0 – 3	0 – 15	0 – 3
Group 3: $\overline{\text{csp0}}$, $\overline{\text{csp1}}$, $\overline{\text{csp2}}$, $\overline{\text{csp3}}$	0 – 3	0 – 15	0 – 3
Group 4: $\overline{\text{csp4}}$, $\overline{\text{csp5}}$, $\overline{\text{csp6}}$, $\overline{\text{csp7}}$	0 – 3	0 – 15	0 – 3
Asynchronous DRAM $\overline{\text{cas}}$ cycle	0 – 3	0 – 3	0 – 3
Asynchronous DRAM $\overline{\text{ras}}$ cycle	0 – 3	0 – 3	–
Asynchronous DRAM $\overline{\text{ras}}$ precharge cycle	–	0 – 3	–

Table 5-5 Possible Wait State Settings

Wait states for groups 1 to 4 are configured in R_WAITSTATES, and asynchronous DRAM wait states are configured in R_DRAM_TIMING. A zero wait state bus cycle lasts 20 ns. Between bursts there are 10 ns. Each wait state adds 10 ns to the bus cycle time. Default after reset is maximum number of wait states.

For Synchronous DRAM banks, the following wait state parameters can be configured:

Parameter	Name	Size (bits)	Description
Master clock select	clk100	1	Synchronous DRAM master clock select. Possible configuration is either 50 MHz or 100 MHz.
$\overline{\text{cas}}$ latency cycles	cl	2	Number of delay cycles from read bank command to valid read data.
$\overline{\text{ras}}$ to $\overline{\text{cas}}$ delay cycles	rcd	2	Number of delay cycles after activate bank command.
$\overline{\text{ras}}$ precharge delay cycles	rp	2	Number of delay cycles after precharge bank command.
Row cycle time	rc	2	Auto refresh cycle time.
Power down exit delay	pde	1	Number of delay cycles from power down exit to new command.

Table 5-6 SDRAM Bank Wait State Parameters That Can Be Configured

Table 5-7 below gives possible Synchronous DRAM wait state settings and corresponding delay cycles for both a 50 and 100 MHz master clock:

Cycle	Wait State Ranges		Delay Cycle Ranges	
	50 MHz	100 MHz	50 MHz	100 MHz
$\overline{\text{cas}}$ latency	1 - 2	0 - 1	2 – 3	2 – 3
$\overline{\text{ras}}$ to $\overline{\text{cas}}$ delay	0 - 3	0 - 3	1 - 4	2 - 5
$\overline{\text{ras}}$ precharge delay	0 - 3	0 - 3	1 - 4	2 - 5
Row cycle time	0 - 3	0 - 3	6 - 9	6 - 9
Power down exit delay	0 - 1	0 - 1	1 - 2	1 - 2

Table 5-7 Possible Wait State Settings and Delay Cycles for Synchronous DRAM

A delay cycle lasts 20 ns when a 50 MHz SDRAM system clock is selected, and 10 ns when a 100 MHz system clock is selected.

5.6 Memory Timing

The timing for the read cycles, both with and without early wait states, is shown in the figure below. This timing diagram is valid for SRAM, flash and peripheral devices.

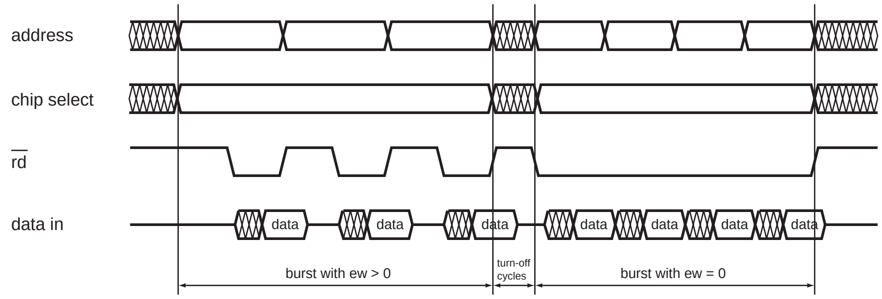


Figure 5-3 Read Cycles for SRAM, Flash Memory and Peripheral Devices.

5.7 Write Modes

5.7.1 Normal and Extended Write Mode

During a normal ETRAX 100LX write cycle, the $\overline{wr0}$ - $\overline{wr3}$ strobes will go high 5 ns before the end of the bus cycle. The write pulses can be extended to the end of the bus cycle. Normal and extended write mode can be configured individually for each of the groups, 1 to 4, of the memory banks in R_BUS_CONFIG. Asynchronous DRAM is configured in R_DRAM_TIMING.

5.7.2 Bytewise and Common Write Enable Mode

The memory banks in group 2, the SRAM group, can be configured with R_BUS_CONFIG to use either four bytewise write enable strobes or one common write enable strobe. In bytewise write enable mode, bwe, four write strobes are available, one for each byte in the data bus. In common write enable mode, cwe, one common write enable strobe and four byte enable strobes are available. The byte enable strobes are active during both read and write cycles. In 32-bit cwe mode, one of the byte enable strobes is output on the unused address bit 1.

16/32-bit bwe	32-bit cwe	16-bit cwe
$\overline{wr0}$	$\overline{be0}$	$\overline{be0}$
$\overline{wr1}$	$\overline{be1}$	$\overline{be1}$
$\overline{wr2}$	$\overline{be2}$	-
$\overline{wr3}$	$\overline{we}$	$\overline{we}$
a1	$\overline{be3}$	a1

Table 5-8 Pin Assignment for Bytewise and Common Write Enable Modes

Note 5: For information regarding actual pin assignments see chapter *19 Electrical Information.*

The byte enable strobes, $\overline{be0}$ - $\overline{be3}$, have the same timing as the address bus. The write enable strobe, $\overline{we}$, has the same timing as the write strobes, $\overline{wr0}$ - $\overline{wr3}$.

5.8 External Interrupt Acknowledge

If the external interrupt is configured for an external vector number, an interrupt acknowledge cycle will occur when the interrupt is granted. During the interrupt acknowledge cycle, an 8-bit vector number is read on the low byte of the data bus. The cycle is indicated by the inta signal going low, and rd and wr are high during the cycle. The address and chip selects are undefined. Maximum timing is used for the inta cycle, regardless of setting: ew = 3, lw = 15, and zw = 3.

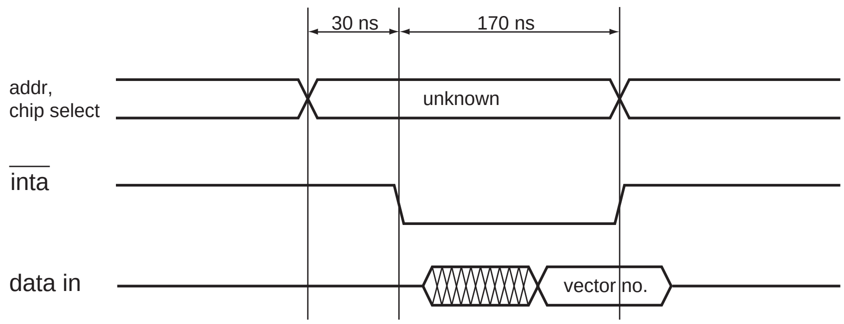


Figure 5-4 External inta Cycle

Note 6: For more information see chapter *19 Electrical Information.*

5.9 Access to Internal I/O

Data read from or written to an internal I/O unit is present on the data bus for 2 clock cycles (i.e. 20 ns). The rd pulse is active for 20 ns and the wr pulses are active for 10 ns. All chip selects are inactive.

5.10 Wait Input and Bus Cycle Rerun

An external wait pin (wait) is provided, and can be used by external devices to insert extra wait states. The wait input is fully synchronized, and wait is sampled 3 clock cycles (30 ns) before the end of the bus cycle. As a result, the use of the wait input requires that the number of internal wait states (ew + lw) for the memory area in question to be set to 3 or more.

If the wait input is active for too long, it can cause overrun/underrun errors (e.g. in the network interface). The maximum allowed active time is, therefore, limited depending on the application.

For very slow external units it is possible to make a bus cycle rerun. However, this only applies for non-cacheable CPU accesses and is not allowed to be used on other types of cycles, like DMA cycles etc. The rerun input is sampled at the rising edge of the wait signal. If the rerun input is sampled low, the CPU will rerun the bus cycle.

5.11 DRAM Interfaces

The DRAM interface can be configured to use *Asynchronous DRAMs* (EDO or Fast Page Mode), *Synchronous DRAMs* (SDRAM), or *Double Data Rate Synchronous DRAMs* (DDR SDRAM).

Selection between asynchronous and synchronous modes is made by the sdram mode bit in the R_DRAM_TIMING register. When this mode bit is not set (sdram = 0), the asynchronous DRAM mode is enabled and the registers: R_DRAM_TIMING and R_DRAM_CONFIG are used for configuration.

When the sdram mode bit is set (sdram = 1) in R_DRAM_TIMING, the SDRAM mode is enabled and R_SDRAM_TIMING and R_SDRAM_CONFIG are used for configuration.

ASYNC DRAM	SDRAM	DDR SDRAM
a25	sdram_ras	sdram_ras
a24	sdram_cas	sdram_cas
a23	sdram_we	sdram_we
ras0	csd0	csd0
ras1	csd1	csd1
ras2	clk	clk
ras3	cke	cke
cas<7.0>	dqm<7.0>	dqm<7.0>
dramwe	-	dqs

Table 5-9 Pin Assignment for Asynchronous and Synchronous DRAM modes

Note 7: For information regarding actual pin assignments see *19 Electrical Information*.

The DRAM interface supports up to four banks of asynchronous DRAM chips and eight banks of Synchronous DRAM chips without external logic. 32-bit and 64-bit wide DRAM modules are supported as well as separate DRAM chips. DRAM bus width can be configured to 16 or 32 bits.

All DRAM accesses are performed in bursts. When the first access in a series of bursts is performed, the internal address is shifted down so that the row address appears on the lower pins of the address bus. The row address is stored in an internal register in ETRAX 100LX for later reference. When the row address portion of the cycle is finished, the column address is put on the address bus.

For every new burst, the row address is compared with the previously used row address that is stored in an internal register. If the row addresses are identical, the row address portion of the cycle can be left out which saves that time that would have been needed for the row access.

5.12 Asynchronous DRAM Interface

The ETRAX 100LX supports both Fast Page Mode and Hyper Page Mode (EDO) asynchronous DRAM chips. The timing diagram for a Fast Page Mode read cycle is shown in figure 5-5 below:

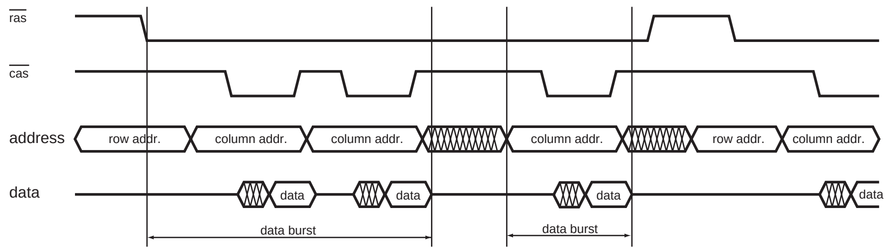


Figure 5-5 Timing Diagram for Fast Page Mode Read Cycles

If EDO DRAM chips are used, ETRAX 100LX uses the $\overline{\text{ras}}$ signal or the $\overline{\text{dramwe}}$ signal to tell the DRAM to release the data bus. The DRAM will release the data bus either when the $\overline{\text{ras}}$ signal goes high, or when the $\overline{\text{dramwe}}$ signal goes low.

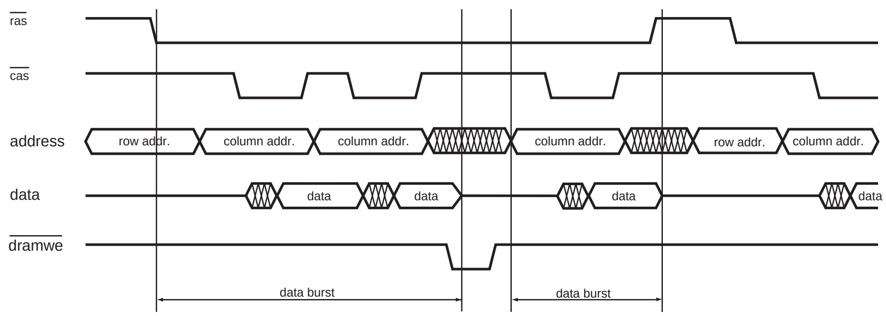


Figure 5-6 Timing Diagram for EDO Mode Read Cycles

5.12.1 Connecting the Asynchronous DRAM Banks

Four asynchronous DRAM banks are combined into two groups with two banks in each group. Both banks in one group must have the same number of column address bits.

When accessing these DRAM banks, the $\overline{\text{cas}}$ signals can be used either as bank strobes (bankwise $\overline{\text{cas}}$ mode), or as byte strobes (bytewise $\overline{\text{cas}}$ mode). One mode is used throughout a single application.

Bytewise $\overline{\text{cas}}$ mode is illustrated in figure 5-7 below. Only one $\overline{\text{ras}}$ signal at a time will be active in each group. The $\overline{\text{cas}}$ signals are used to select the different bytes within the word or dword.

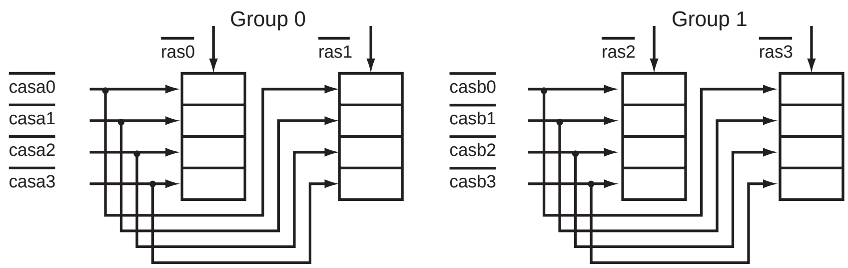


Figure 5-7 Bytewise cas Mode.

In a design using bankwise cas mode, as in figure 5-8, all the ras signals can be active simultaneously. casa decides which bank to access. If bytewise access is required, the byte cas signals are generated by gating casb, used as byte enables, and casa as shown in figure 5-9.

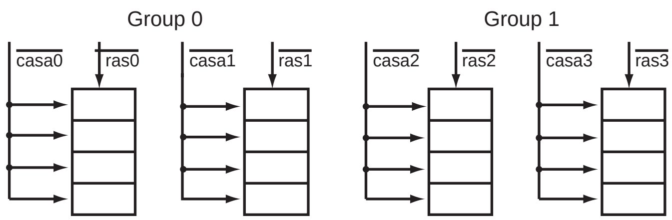


Figure 5-8 Bankwise Accessing of Asynchronous DRAM Modules.

Bankwise cas mode can be used without the cas gating if the software fulfills the following requirements:

- CPU accesses to DRAM area are always cached.
- All DMA descriptors are 32-bit aligned.
- A DMA data buffer and program code/data, or two different DMA data buffers, are not merged within the same 32-bit aligned dword.

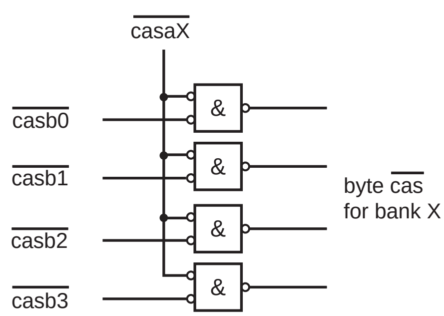


Figure 5-9 Generating Byte cas.

When using 64-bit wide asynchronous DRAM modules, one $\overline{\text{ras}}$ signal is assigned to each bank and this signal controls all 64 bits. Only one $\overline{\text{ras}}$ is active at a time. The $\overline{\text{cas}}$ signals are used to select the different bytes within the word, see figure 5-10.

Upper and lower 32 bits are also tied together. In other words, bit 0 and bit 32 (bit 1 and bit 33, bit 2 and bit 34, etc.) are tied together, and one at a time they drive the databus.

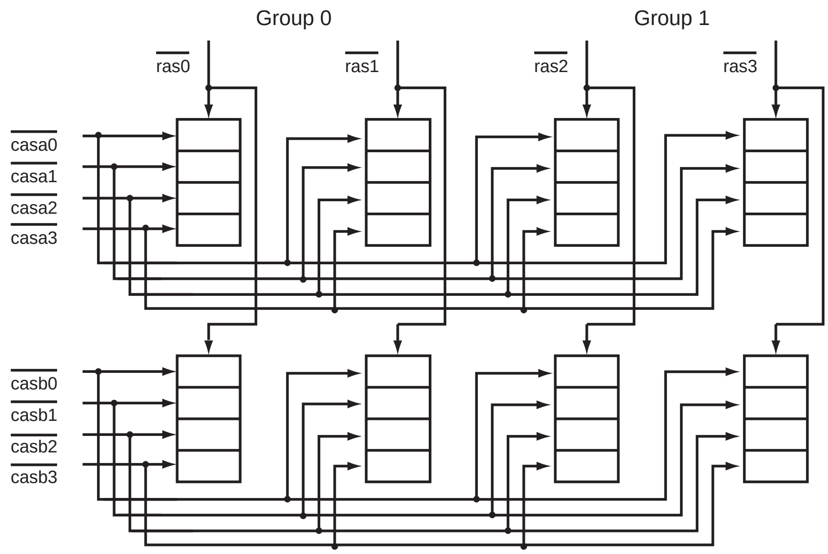


Figure 5-10 64-bit Wide Asynchronous DRAM Modules, All $\overline{\text{cas}}$ Signals Used For Selecting Bytes.

In bankwise mode with 64-bit wide modules, the $\overline{\text{cas}}$ signals are used to activate 32 bits at a time as shown in figure 5-11. In this case, all $\overline{\text{ras}}$ signals can be active at the same time, but it is not possible to select the separate bytes in the DRAM.

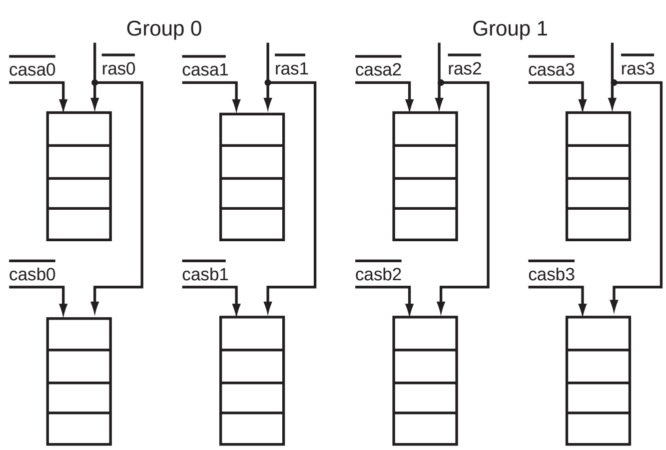


Figure 5-11 Bankwise Accessing, 64-bit Wide Asynchronous DRAM Modules.

5.12.2 Asynchronous DRAM Bank Configuration

Four asynchronous DRAM banks are combined into two groups with banks 0 and 1 in group 0, and banks 2 and 3 in group 1. The two banks in each group always share the same configuration from R_DRAM_CONFIG.

Common to all banks, the following configurations are available:

Width (1 bit)

Selects 16- or 32-bit DRAM bus width.

0	16-bit DRAM bus width.
1	32-bit DRAM bus width.

EDO or Fast page mode (1 bit)

0	Fast page mode.
1	EDO mode.

$\overline{\text{cas}}$ organization (1 bit)

0	The $\overline{\text{cas}}$ outputs are used for selecting bytes as in figure 5-7 or, if wide module mode is selected, as in figure 5-10.
1	One $\overline{\text{cas}}$ output for each bank, see figure 5-8 and figure 5-9, or if wide module mode is selected, two $\overline{\text{cas}}$ outputs per bank as in figure 5-11.

Group select mode (5 bits)

The group select mode determines how to select a group of DRAM banks:

Value	Select
0	Always select group 0.
1	Always select group 1.
2-8	Reserved.
9-29	The internal address bit number corresponding to the select mode value is used to select group.
30-31	Reserved.

The following configurations are available for each group individually:

Row Address Shift (3 bits)

During the $\overline{\text{ras}}$ portion of the cycle, the row portion of the internal address is shifted down to the lower address outputs to which the asynchronous DRAM address pins are connected. The value given decides how many steps the address bits are shifted:

Value	Shift
0	Internal address bits 29 - 9 shifted down to pins A21 - A1.
:	:
:	:
7	Internal address bits 29 - 16 are shifted down to pins A14 -A1, and pins A21 - A15 are set to 0 (zero).

(The internal address bits 25 - 22 are output on pins A25 - A22 during $\overline{\text{ras}}$ cycles.)

To accomplish the multiplexing of the row address and the column address on the address bus pins, the row address is shifted down to the lower pins of the bus during the $\overline{\text{ras}}$ portion of the bus cycle.

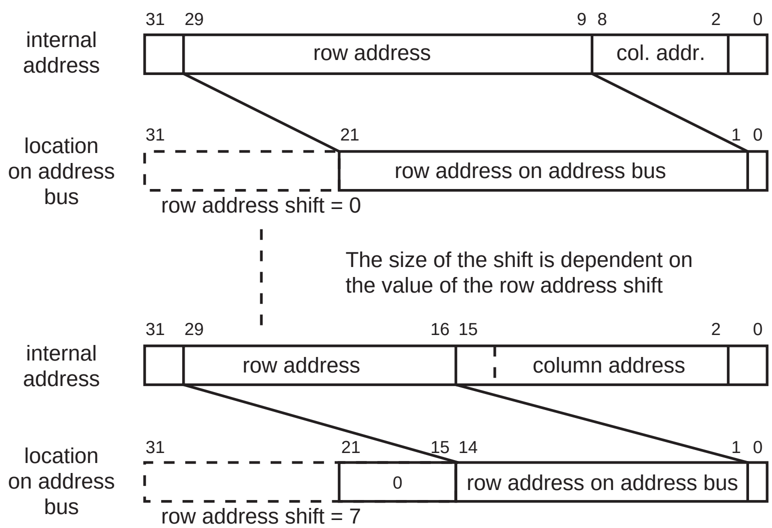


Figure 5-12 Row Address Shifting During the $\overline{\text{ras}}$ Portion of the Bus Cycle

Column Address Range (3 bits)

The column address range determines how many address bits that are used in the column address.
The ranges selected for the column address and the row address may overlap by one bit. When the row address has been shifted down, this bit corresponds to bit 1 on the address bus, which is not used in the 32-bit address. Consequently, the overlapping bit is part of the column address.
There may also be a gap between the row address and column address parts, which

are not used by the asynchronous DRAM bank. The bits in the gap can be used to select the bank and/or group.
In wide module mode, the selection between casa and casb is done by the highest address bit given by the column address range.The column address range should, in this case, be set to one higher than the actual highest column address bit to the DRAM.

Value	Column Address Bits
0	Up to and including address bit 8.
:	:
:	:
7	Up to and including address bit 15.

Bank Decode Mode (5 bits)

The bank decode mode determines how to select between the two banks in a group:

Value	Select
0	Always select bank 0 (group 0) or bank 2 (group 1).
1	Always select bank 1 (group 0) or bank 3 (group 1).
2-8	Reserved.
9-29	The internal address bit number corresponding to the decode mode value is used to select bank.
30-31	Reserved.

Wide Module Mode (1 bit)

This mode supports 64-bit wide asynchronous DRAM modules where all 64 bits are controlled by the same ras signal. Both casa and casb are used within both groups of DRAM banks.
If bankwise cas mode is combined with the wide module mode in any of the two groups, none of the groups can use casb as byte enables, see figure 5-11.

Value:	Select Mode:
0	Normal mode.
1	Wide module mode.

5.13 Synchronous DRAM Interface

The SDRAM interface can be configured to use a master clock of either 50 or 100 MHz. In 100MHz mode, the interface can also work as a DDR SDRAM interface. The maximum internal bus speed is always 50MHz regardless of the SDRAM master clock selection.

There is support for two groups of SDRAM banks. A group is the set of SDRAM banks that are collectively either 16, 32, or 64 data bits wide. The groups are controlled by the chip selects, csd0 and csd1. Each group can be configured to use either two-bank or four-bank chips.

The timing for a 50 MHz SDRAM read cycle is shown in figure 5-13:

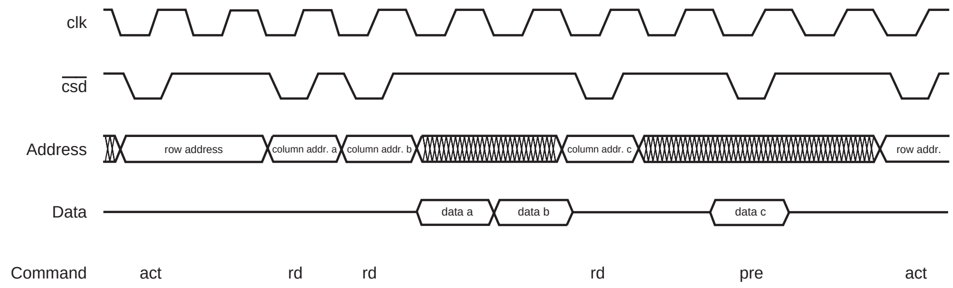


Figure 5-13 Timing Diagram for 50 MHz Read Cycles with $\overline{cas}$ Latency and $\overline{ras}$ Precharge Delay of 2 Delay Cycles

In 100MHz mode commands are only issued once every two clock cycles in order to maintain the internal bus speed of 50MHz.

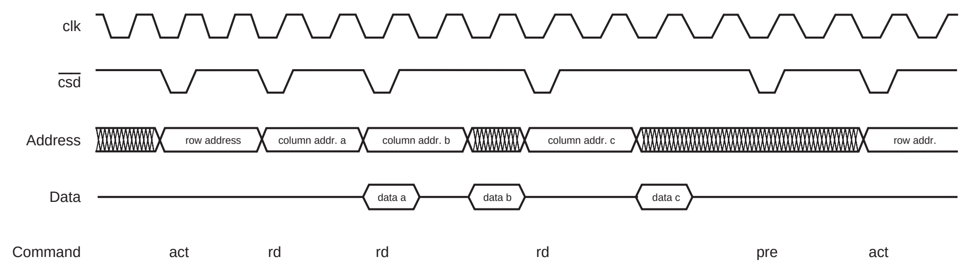


Figure 5-14 Timing Diagram for 100 MHz Read Cycles with $\overline{cas}$ Latency and $\overline{ras}$ Precharge Delay of 2 Delay Cycles

In DDR mode, a bi-directional data strobe, *dqs*, is used to qualify the data bus. The minimal burst length supported by DDR devices are two words and the second word is, therefore, masked during writes by the data mask, *dqm*, signals. Write latency for DDR devices is always set to one cycle.

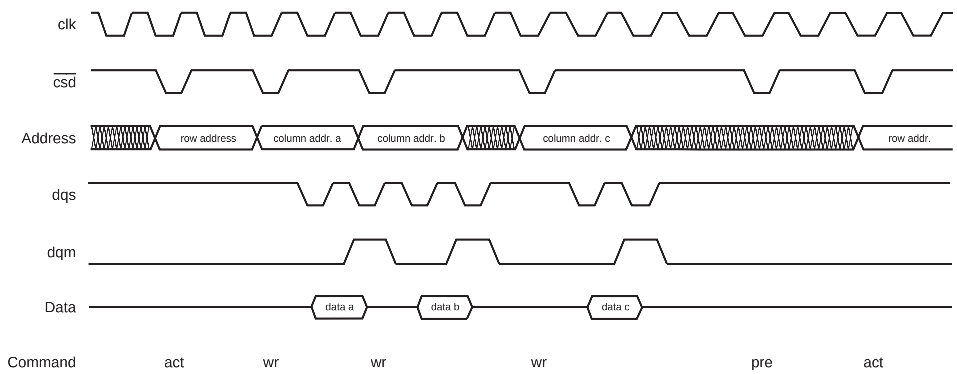


Figure 5-15 Timing Diagram for 100 MHz DDR Write Cycles

5.13.1 Power up and initialization

SDRAMs have an internal configuration register that must be written during initialization with the *mode register set* (mrs) command. The SDRAM must be in idle mode and the mrs command should, therefore, be preceded by an auto refresh cycle to make sure that the banks are pre charged. The mrs command is issued by writing to the cmd field in R_SDRAM_TIMING. During the mrs cycle, the mrs data field in R_SDRAM_TIMING is output on address bits 15 - 0. All bits in the mrs field should normally be set to zero except for the $\overline{\text{cas}}$ latency select bits.

It is possible to issue commands manually to the SDRAM by writing to the command field in R_SDRAM_TIMING. The possible commands are:

- precharge all banks
- auto refresh
- mode register set
- nop

All commands have to be followed by a nop command before a new command can be issued. Note that at least five CPU nop instructions should be inserted between each write to R_SDRAM_TIMING.

The manual commands should only be used during power up and initialization of the SDRAM banks. A typical initialization sequence is shown below:

1 Configure the banks by writing to R_SDRAM_CONFIG.
2 Configure the waitstate parameters and enable the master clock by writing to R_SDRAM_TIMING.
3 Wait for 200us.
4 Issue precharge all banks command by writing to the cmd field in R_SDRAM_TIMING.
5 Wait five CPU nop cycles.
6 Issue nop command by writing to the cmd field in R_SDRAM_TIMING.
7 Wait five CPU nop cycles.

Next, perform the following sequence eight times:

1 Issue auto refresh command by writing to the cmd field in R_SDRAM_TIMING.
2 Wait five CPU nop cycles.
3 Issue nop command by writing to the cmd field in R_SDRAM_TIMING.
4 Wait five CPU nop cycles.

Finally,

1 Issue mode register set command by writing to the cmd and mrs_data fields in R_SDRAM_TIMING.
2 Wait five CPU nop cycles.
3 Issue nop command by writing to the cmd field in R_SDRAM_TIMING.
4 Wait five CPU nop cycles.

5.13.2 Power save mode

It is possible to run the SDRAM interface in the Power save mode. In this mode, the SDRAM banks will enter Power save mode immediately after each auto refresh cycle. This is done by deasserting the cke signal. The SDRAM banks will remain in Power save mode until the next SDRAM bus request, or until they have to be refreshed again. There is a penalty of one or two delay cycles for each Power save mode exit. The Power save mode exit delay can be configured in R_SDRAM_TIMING.

5.13.3 100 MHz mode

It is possible to select a100 MHz system clock together with both SDRAMs and DDR SDRAMs, but this will not give double the performance compared to the 50 MHz clock. Commands will only be issued every two cycles due to the internal 50 MHz bus speed. There will, however, be a speed-up since $\overline{\text{ras}}$ and $\overline{\text{cas}}$ latency will be shorter.

5.13.4 DDR mode

The DDR mode must be used with a 100 MHz system clock due to an internal DLL in the DDR SDRAM chips. Since the DDR SDRAMs only use the dqm signals as a data mask during write operations, 64-bit modules with one common chip select for all eight bytes can not be used.

5.13.5 Connecting the Synchronous DRAM banks

The Synchronous DRAM banks are combined into two groups where each group is controlled by a separate chip select signal.

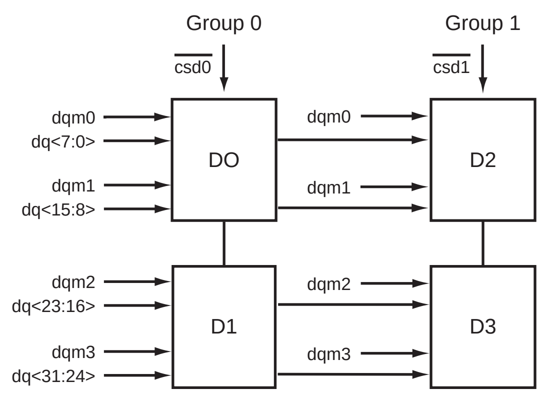


Figure 5-16 32-bit SDRAM Connection Comprising Four 16-bit SDRAM Devices.

When using 64-bit wide DRAM modules, one chip select is assigned to each group and controls all 64-bits. The eight dqm signals are used to select the different bytes within the word.

Upper and lower 32 bits are tied together. In other words, bit 0 and bit 32 (bit 1 and bit 33, bit 2 and bit 34, etc.) are tied together, and one at a time they drive the databus:

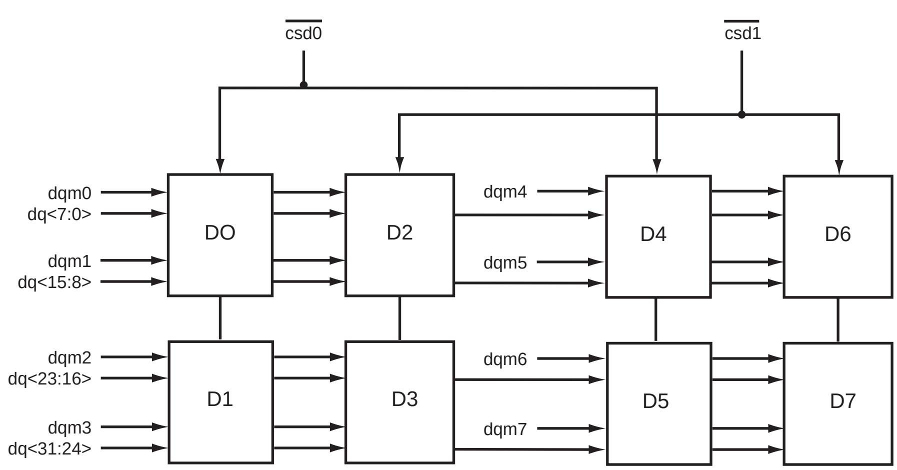


Figure 5-17 64-bit Wide SDRAM Module Comprising Eight 16-bit SDRAM Devices.

5.13.6 Synchronous DRAM Bank Configuration

SDRAM banks are combined into two groups where each group can use either two-bank or four-bank chips. The following configurations are common for all banks:

Width (1 bit)

Width selects either a 16- or 32-bit DRAM bus width.

Value	Select
0	16-bit DRAM bus width.
1	32-bit DRAM bus width.

Group Select Mode (5 bits)

The group select mode determines how to select a group of DRAM banks. When using only one group, the group select mode value must be set to either 0 or 1.

Value	Select
0	Always select group 0.
1	Always select group 1.
2-8	Reserved.
9-29	The internal address bit number corresponding to the select mode value is used to select group.
30-31	Reserved.

Row Address Shift (3 bits)

During activate bank commands, the row portion and the bank select bits are shifted down to the lower address outputs to which the DRAM address pins are connected. During precharge bank, read bank, and write bank commands, only the bank select bits are shifted down to the lower address outputs. When using two groups, both groups must have the same row address shift value.

The value given decides how many steps the address bus is shifted

Value	Shift
0	Internal address bits 29 - 9 are shifted down to pins A21 - A1.
:	:
:	:
7	Internal address bits 29 - 16 are shifted down to pins A14 - A1, and pins A21 - A15 are set to 0(zero).

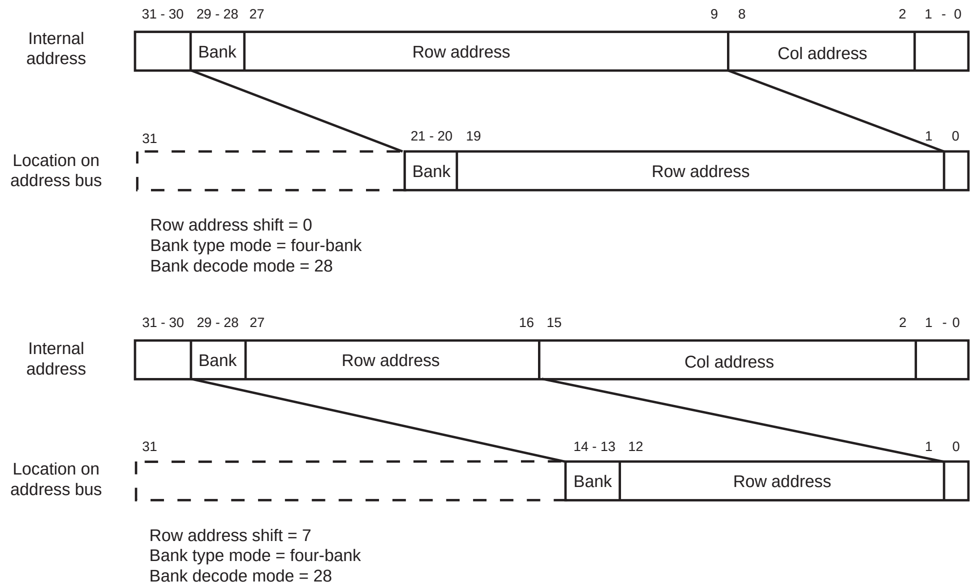


Figure 5-18 SDRAM Address Bus Shift During Activate Bank Commands

Column Address Range (3 bits)

The column address range determines how many address bits that are used in the column address.
The ranges selected for the column address and the row address may overlap by one bit. When the row address has been shifted down, this bit corresponds to bit 1 on the address bus, which is not used in the 32-bit address. Consequently, the overlapping bit is part of the column address.
There may also be a gap between the row address and column address parts, which are not used by the asynchronous DRAM chip. One bit in the gap can be used to select the group.
In wide module mode, the selection between $\overline{\text{casa}}$ and $\overline{\text{casb}}$ is done by the highest address bit given by the column address range.The column address range should, in this case, be set to one higher than the actual highest column address bit to the DRAM.

Address bit a10 on SDRAM chips is used to select auto precharging during read and write commands and is not used as a column address. In 32-bit mode, a10 will correspond to a12 on the ETRAX 100LX, and in 16-bit mode, a10 will correspond to a11. a10 is never used as an address bit during read and write commands. In configurations with more than 10 column addresses, the address continues on the next higher address bit after a10. When using two groups, both groups must have the same column address range value.

Value	Column Address Bits
0	Up to and including address bit 8.
:	:
:	:
7	Up to and including address bit 15.

The following configurations are available for each group individually:

Bank Type Mode (1 bit)

Bank type mode selects either the two-bank or four-bank mode:

Value	Select
0	Two-bank mode is selected.
1	Four-bank mode is selected.

Bank Decode Mode (5 bits)

The bank decode mode determines how to select between bank 0 and 1 in a group. In the four-bank mode, bank 2 and 3 will selected by the next higher order address bit. The bank select bit or bits are shifted down to the lower address outputs according to the row address shift value.

Value	Column Address Bits
0 - 8	Reserved.
9 - 29	Internal address bit number corresponding to the decode mode value is used to select between bank 0 and 1.
30 - 31	Reserved.

Wide Module Mode (1 bit)

This mode supports 64-bit wide DRAM modules where all 64-bits are controlled by the same chip select signal. dqm0 to dqm7 are used to control the individual bytes within both groups of DRAM banks.

Value:	Select Mode:
0	Normal mode
1	Wide module mode

6 BOOTSTRAP METHODS

6.1 Bootstrap Methods

There are four different methods to bootstrap the ETRAX 100LX. They are presented in table 6-1 below.

The cache is always initialized, regardless of the bootstrap method. The bus status pins bs3, bs2, bs1, and bs0 are sampled upon reset going high, and their values are stored to the internal register R_BUS_STATUS. The values of the bus status pins bs2 and bs1 determine which bootstrap method to use.

Values on pins bs2 and bs1 at reset	Bootstrap method	Description
00	Normal bootstrap	Execution starts at address 0x80000002.
01	Serial bootstrap	Serial port 0 is used, configured at 9600bps, 8 bits without parity, one start and one stop bit).
10	Network bootstrap	The network bootstrap code is received in an Ethernet packet through the SNI or MII interface.
11	Parallel bootstrap	Parallel port 0 is used.

Table 6-1 Overview of the ETRAX 100LX Bootstrap Methods

6.1.1 Normal Bootstrap

Execution starts at address 0x80000002.

6.1.2 Serial Bootstrap

Serial port 0 is used, and is configured at 9600bps, 8 bits without parity, one start and one stop bit. A total of 784 bytes will be received. This data is stored in the cache, mapped to the address 0x380000F0, where execution then starts.

6.1.3 Network Bootstrap

The network bootstrap code is received in an Ethernet packet, as shown in table 6-2, through the SNI or MII interface. When the network bootstrap is used, the packet received must be an Ethernet packet (formatted to the IEEE 802.3 standard).

Packet byte nr.	Content (hex)	Description
0 - 5	01 40 8c 00 01 00	Destination address. This is a multicast address within the Axis Ethernet address block. This address is fixed.
6 - 11	XX XX XX XX XX XX	Source address. The address of the host transmitting the bootstrap packet. This address is not checked.
12 - 13	type-length (2bytes)	This is currently not checked but it is recommended that the contents follow the 802.3 standard.

Table 6-2 Network Bootstrap Packet Header

Packet byte nr.	Content (hex)	Description
14 - 21	AA AA 03 00 40 8C 88 56	A SNAP header featuring the Axis vendor code (same as the Ethernet address block). This is the Axis ether-type specifically assigned for this purpose. This is currently not checked but is strongly recommended for network interoperability.
22 - 25	FF FF FF FF	A tag signalling this packet as a bootstrap datagram. This is currently not checked, but is recommended for re-use of the Axis SNAP header.
26 - 29	00 00 00 00	The bootstrap packet sequence number. The number must consist of only zeros in the first packet in the bootstrap sequence. This field is fixed.

Table 6-2 Network Bootstrap Packet Header

After this header, the network bootstrap code starts.The downloaded program can be up to 1484 bytes in this package, and will be stored in the cache. The first byte of the destination address is mapped to address 0x380000E6, and execution will start at address 0x380000F4.

The SNI and MII interfaces are selected by the value of the MDIO pin at start-up:

MDIO pin value	Interface
0	SNI
1	MII

Table 6-3 Value of the MDIO Pin at Start-up

6.1.4 Parallel Bootstrap

Parallel port 0 is used. A total of 784 bytes will be received. The parallel port will immediately start in ECP reverse mode for the transfer of this data to the cache where it is stored. The data is mapped to address 0x380000F0, where execution then starts. IEEE 1284 negotiation phase is not performed.

7 DMA

DMA in the ETRAX 100LX provides a high data transfer rate to and from the internal peripheral interfaces, or from one location to another location in the external memory. It consists of ten DMA channels with five in each direction. The ten DMA channels are served by a DMA controller, which takes care of the data flow between the channels and the external memory. In addition, there are two external DMA channels (See *7.8 External DMA Channels*).

7.1 DMA Operation

7.1.1 Overview of the ETRAX 100LX DMA Architecture

A simplified architecture overview of DMA in the ETRAX 100LX, as well as a schematic flow of data is shown in figure 7-1 below.

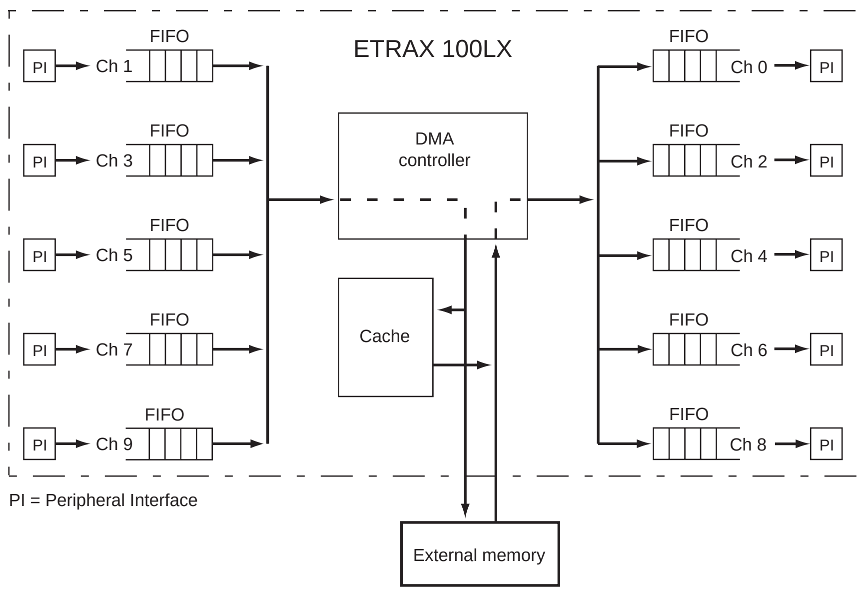


Figure 7-1 DMA Internal Overview

There are five input channels (1, 3, 5, 7 and 9), and five corresponding output channels (0, 2, 4, 6 and 8). All input and output channels have a FIFO buffer.

7.1.2 Data Transfer

There are three cases of data transference from:

1. A peripheral interface to external memory
2. External memory to a peripheral interface
3. One memory location to another memory location

Apart from these three cases of data transference, ETRAX 100LX DMA has two external DMA channels. See *7.8 External DMA Channels*.

Data Transfer from a Peripheral Interface to External Memory

When data is to be transferred from one peripheral interface, it is first stored in a 64 byte FIFO buffer. When this buffer fills up to half its size (i.e. 32 bytes), the DMA controller is notified, and the data in the FIFO is transferred to external memory. If one of these memory addresses is present in the internal cache memory, the data is also stored in the internal cache memory. The transfer of data from a FIFO to the memory is done in bursts, and the length of these bursts is either 16 or 32 bytes as chosen by the software programmer in the register R_BUS_CONFIG.

Data Transfer from External Memory to a Peripheral Interface

When the DMA controller receives notice that a FIFO buffer is becoming half empty (i.e. less than or equal to 32 bytes), it begins a transfer of data from the external memory. When data is transferred from the external memory, it is read in bursts of 16 or 32 bytes as chosen by the software programmer in R_BUS_CONFIG. If the data in one of these memory addresses is present in the internal cache memory, the data is read from the internal cache memory.

Data Transfer from One Memory Location to Another Memory Location

In order to increase the performance of DMA for this type of data transfer, there are two channels specifically designed for this task: channels 6 and 7. The FIFO buffers for channels 6 and 7 can be set to connect directly to each other in the register R_GEN_CONFIG.

As with previous transfers, the data is read from and written to external memory when the FIFOs are either half empty or half full. If one of the memory addresses, where the data is located, is present in the internal cache memory, the data is read from or written to the internal cache memory. The data transfer to and from the FIFOs is done in bursts of 16 or 32 bytes as chosen by the software programmer in R_BUS_CONFIG.

7.2 The DMA Channels

There are only a limited number of combinations for the ten DMA channels which can be used for interconnection between the internal peripheral interfaces. The reason for this limitation is that some of the internal peripheral interfaces are multiplexed on the same package pins. Figures 7-2 and 7-3 below show how each peripheral interface is multiplexed on the DMA channels.

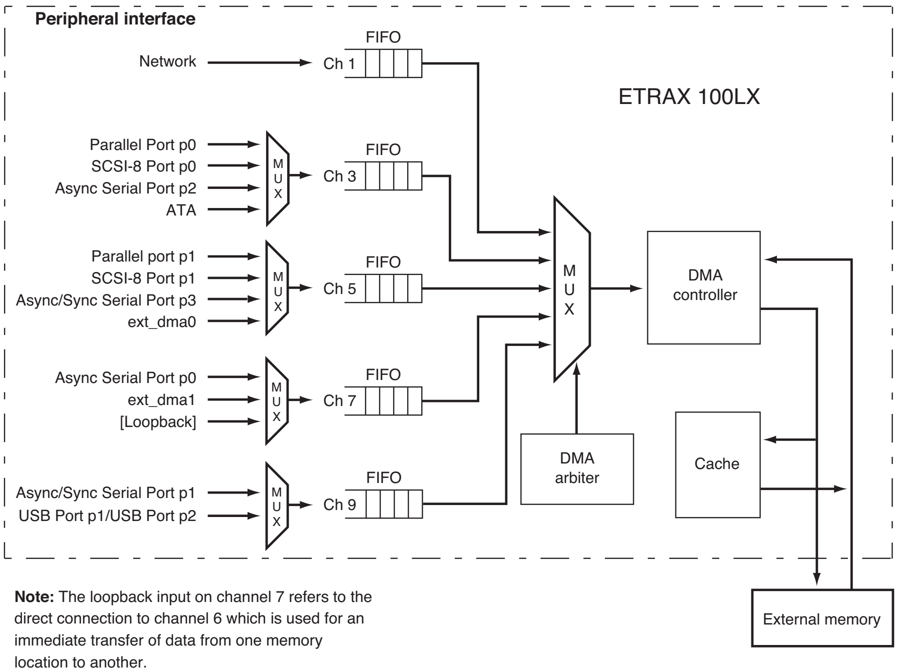
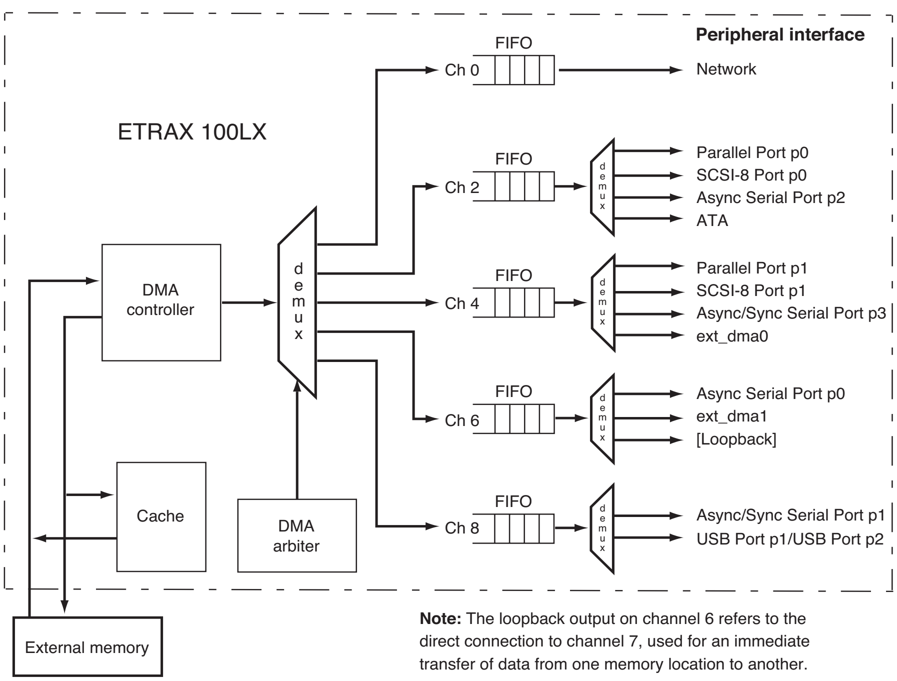


Figure 7-2 DMA Input Channels



Note: The loopback output on channel 6 refers to the direct connection to channel 7, used for an immediate transfer of data from one memory location to another.

Figure 7-3   DMA Output Channels

The choice of which interface to be used is defined in the register R_GEN_CONFIG. The channels can be configured as shown in table 7-1 below:

DMA channel	I/O system(s) available				Direction	FIFO buffer (bytes)	Priority
0	Network				out	64	Highest
1	Network				in	64	
2	Parallel Port p0	SCSI-8 Port p0	Async serial Port p2	ATA	out	64	
3	Parallel Port p0	SCSI-8 Port p0	Async serial Port p2	ATA	in	64	
4	Parallel Port p1	SCSI-8 Port p1	Async/Sync serial Port p3	ext_dma0 (note 1)	out	64	
5	Parallel Port p1	SCSI-8 Port p1	Async/Sync serial Port p3	ext_dma0 (note 1)	in	64	
6	Async Serial Port p0	ext_dma1 (note 1)	Memory transfer (note 2)		out	64	
7	Async Serial Port p0	ext_dma1 (note 1)	Memory transfer (note 2)		in	64	
8	Async/Sync Serial Port p1	USB Port p1 & p2 (note 3)			out	64	
9	Async/Sync serial Port p1	USB Port p1 & p2 (note 3)			in	64	Lowest

Table 7-1   DMA Output channels

Note 1: ext_dma0 and ext_dma1 are external DMA channels to be used between the external memory and an external I/O device, see *7.8 External DMA Channels*.

Note 2: Memory-to-memory transfer. Channels 6 to channel 7 can be set for immediate connection, which provides an efficient way of transferring data from one memory location to another memory location.

Note 3: When Channels 8 and 9 are configured for USB, their priority becomes higher than the priority for channel 2 but lower than channel 1 (e.g. Highest: channel 0, 1, 8, 9, 2, 3...Lowest). *See 7.4.2 DMA Linked Lists for USB*

All 10 DMA channels (0 - 9) have a 64 byte FIFO buffer to allow efficient handling of burst mode DRAM access, and software can read the number of bytes in each FIFO buffer. For input channels, software can also flush the FIFO contents to the memory data buffer by forcing an eop (see table 7-2) in the data stream flowing into the FIFO by using the register R_SET_EOP.

7.3 DMA Registers, Linked Lists, and Descriptor Format

7.3.1 DMA Registers

There are a set of DMA registers, one set for each channel, which the DMA controller uses to handle the buffers and to store information about the buffers in the external memory. These registers and a little about of their functions are shown in figure 7-4 below:

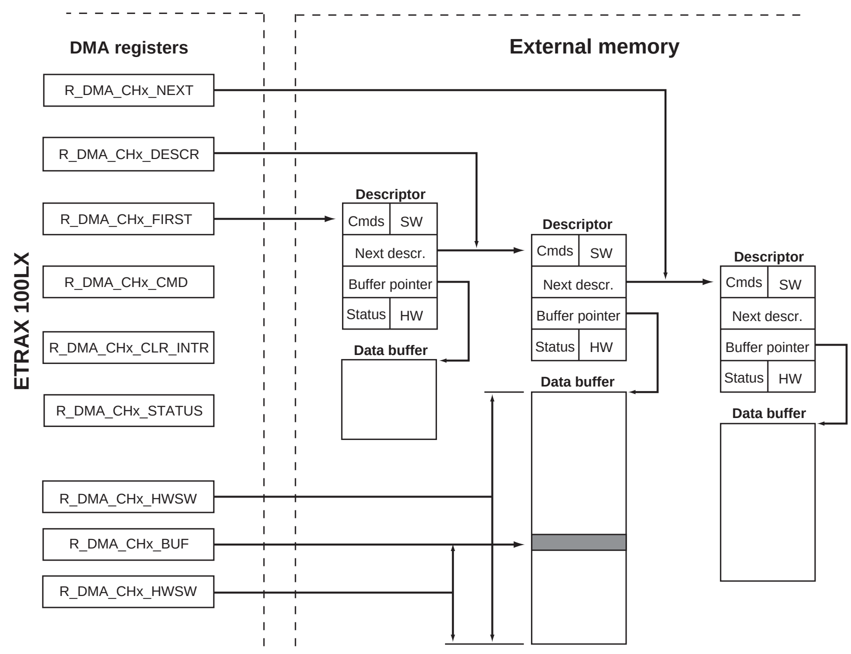


Figure 7-4 DMA Registers and the Structure of the Linked List

The figure above shows the DMA registers and a simplified linked list in the external memory. During normal operation, the only registers accessed by software are:

- R_DMA_CHx_FIRST
- R_DMA_CHx_CLR_INTR
- R_DMA_CHx_CMD

The R_DMA_CHx_FIRST register is used to locate the DMA list. R_DMA_CHx_FIRST points to the first descriptor in the linked list as shown in figure 7-4 above. The R_DMA_CHx_DESCR register contains the current descriptor's address, and the R_DMA_CHx_NEXT contains the address to the next descriptor.

R_DMA_CHx_CLR_INTR is used to clear interrupts. For more details see *7.6 DMA Transfer/Setup Examples.*

R_DMA_CHx_CMD is the command register to control DMA operation, and is used to reset and start DMA transfers. When a command is completed, the DMA channel sets R_DMA_CHx_CMD to hold (0) and stops.

The R_DMA_CHx_CMD register has five commands:

- Hold
- Start
- Restart
- Continue
- Reset

The hold command, which is completed immediately, holds the DMA channel in its current state. The hold command will fail, however, if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by the fact that the DMA channel reaches end-of-list so that R_DMA_CHx_FIRST is zero. A failed hold command is also indicated if the DMA channel receives stop-from-io, and the stop bit is set in the descriptor at R_DMA_CHx_DESCR.

When the start command is given, the DMA channel starts processing the list at R_DMA_CHx_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored.

The restart command restarts the DMA channel after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CHx_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored (See note 4).

A continue command tells the DMA channel to continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. (See note 4)

The reset command resets the DMA channel and its FIFOs. When the channel has won arbitration, the reset command takes 100 ns to complete.

Note 4: The restart and continue commands have the same value (3).

Three registers manage the actual storage of data in the buffers:

- The R_DMA_CHx_BUF register: pointer to the next (byte) position in the data buffer
- The R_DMA_CHx_HWSW register (sw field): gives the total length (in bytes) of the data buffer
- The R_DMA_CHx_HWSW register (hw field): gives the number of bytes left in the data buffer.

7.3.2 DMA Linked Lists

ETRAX 100LX DMA stores data in the external memory, in buffers linked together with the use of a list descriptor. Each list descriptor contains a number of data fields, which tell the DMA controller where to find the next descriptor in the list.

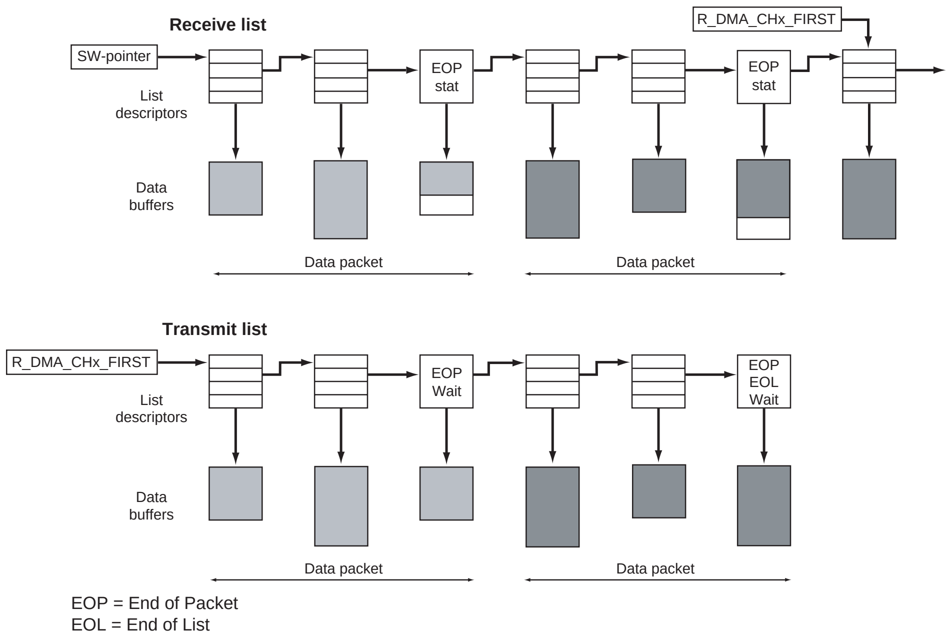


Figure 7-5 ETRAX 100LX Buffer Structure

The list descriptors also contain information about the location and size of the data buffer as well as other status information. This organization of data storage in memory has the advantage of having efficient memory management, and a very flexible structure.

DMA descriptors and data buffers do not have any alignment restrictions, except for the USB EP descriptor which must be 32-bit aligned (See section 7.4.3). However, performance improves if data and DMA descriptors are 32-bit and cacheline aligned.

For a more detailed overview of the list descriptor see *7.3.3 DMA Descriptor Format*.

7.3.3 DMA Descriptor Format

The construction of linked lists is done by defining DMA descriptors. The DMA descriptor consists of four 32-bit fields. The contents of a DMA descriptor is shown in figure 7-6 below:

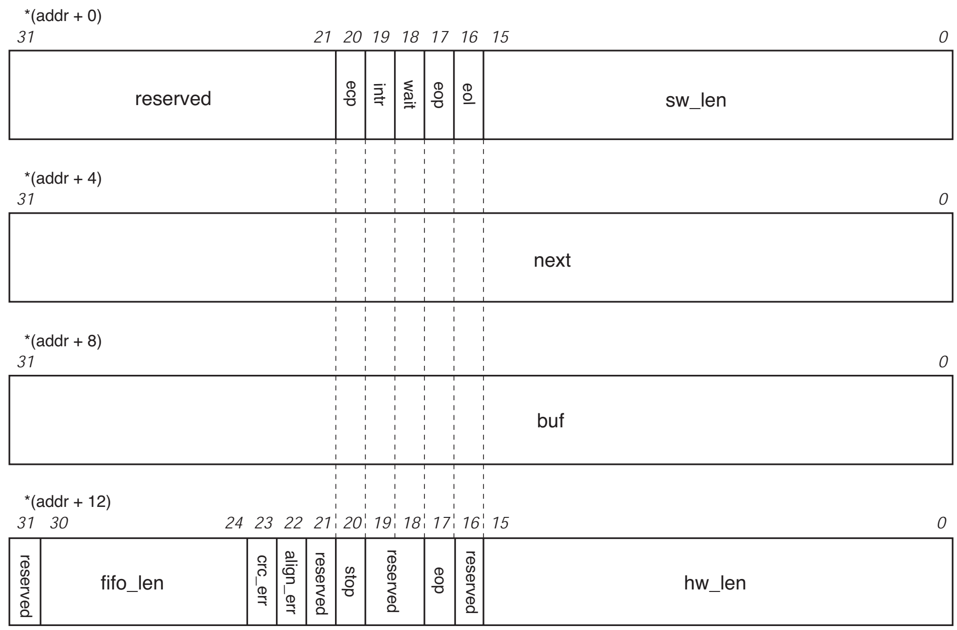


Figure 7-6 The DMA Descriptor Format

The first 32-bit field of the descriptor gives the length of the data buffer sw_len and a number of commands. The second 32-bit field gives the address to the next descriptor in the linked list, and the third 32-bit field gives the address to the data buffer buf. The last 32-bit field contains status information written by the DMA controller.

Table 7-2 describes the contents of the DMA descriptor in more detail:

Bit	Name	Explanation
(addr + 0):		
31-21	reserved	(note 6).
20	ecp_cmd	ECP command used by channel 2 and 4 when connected to a parallel port in ECP mode.
	tx_err	Force transmission error used by channel 0.
19	intr	Generate a descriptor interrupt when advancing to next descriptor.
18	wait	Wait until FIFO is empty before advancing to next descriptor, and also delay descriptor and end-of-packet interrupts until FIFO is empty (output channels only).
17	eop	Last descriptor in a packet (output channels only).
16	eol	Last descriptor in the list.
15-0	sw_len	Length in bytes of data buffer. (If all bits are 0, the length is 2^{16})
(addr + 4):		
31-0	next	Pointer to next descriptor in list (no alignment restrictions). If eol == 1 next is not used.
(addr + 8):		
31-0	buf	Pointer to first byte in data buffer (no alignment restrictions).
(addr + 12): (note 5)		
31	reserved	(note 6)
30-24	fifo_len	If stop == 1; Number of bytes in FIFO (output channels only).
23	crc_err	If eop == 1; Received packet has CRC error, used by channel 1.
22	align_err	If eop == 1; Received packet has alignment error, used by channel 1.
21	reserved	(note 6).
20	stop	Output channel was stopped by I/O interface. Bit 20 and bit 17 are mutually exclusive.
19-18	reserved	(note 6)
17	eop	Last descriptor in a received packet. Bit 20 and bit 17 are mutually exclusive.
16	reserved	(note 6).
15-0	hw_len	If eop == 1; Number of bytes written to the data buffer. (If all bits are 0, the length is 2^{16}) If stop == 1; Number of bytes read from the data buffer. (If all bits are 0, the length is 2^{16}) If eop == 0 && stop == 0; hw_len is not valid. (i.e you have to use sw_len to get the number of transferred bytes)

Table 7-2 DMA Descriptor contents

Note 5: The 32-bit status field is only written if eop or stop is set to one. For software to be able to know that the status field has been written or not, the status field of the descriptor must be cleared to all zeros before the DMA channel is started. The eop bit is written in the last descriptor of a packet.

Note 6: For compatibility with future versions of ETRAX processors, reserved fields must be set to 0 before starting DMA. When read, no assumption can be made about their value.

7.4 DMA Registers, Linked Lists, and Descriptor Format for USB

7.4.1 DMA Registers for USB

Figure 7-7 below shows the DMA registers for USB support in the ETRAX 100LX:

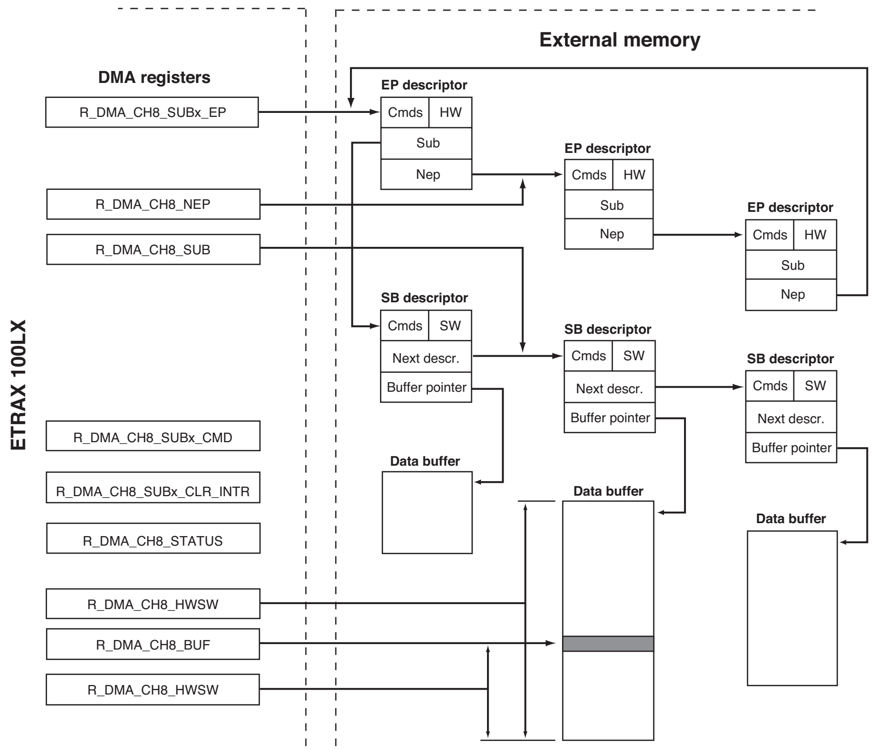


Figure 7-7 DMA Registers for USB

During normal operation, the only registers accessed by software are:

- R_DMA_CH8_SUBx_EP
- R_DMA_CH8_SUBx_CMD
- R_DMA_CH8_SUBx_CLR_INTR

R_DMA_CH8_SUBx_EP is used to locate the circular list of EP descriptors, R_DMA_CH8_SUBx_CMD is used to start the sub channel, and R_DMA_CH8_SUBx_CLR_INTR is used to clear interrupts. For more details see chapter *8 Universal Serial Bus.*

R_DMA_CH8_SUBx_EP points to the current *Endpoint* (EP) descriptor in the EP list. R_DMA_CH8_NEP points to the next EP descriptor in the EP list. R_DMA_CH8_SUB points to the current *Sublist* (SB) descriptor in the current sublist. The subchannel is either enabled or disabled with R_DMA_CH8_SUBx_CMD, and the operation of the sub channels are then controlled by the USB hardware. Interrupts are cleared by writing to R_DMA_CH8_SUBx_CLR_INTR.

7.4.2 DMA Linked Lists for USB

To handle USB in a practical way, DMA channel 8 has extended functionality to handle four sets of two dimensional lists. One set of lists is for each transfer type: control-, interrupt-, isochronous-, and bulk-transfers. When the two dimensional lists are used, the priorities of channels 8 and 9 are higher than channel 2, but lower than channel 1.

The construction of a linked list is done by defining DMA descriptors. There are three descriptor formats in the ETRAX 100LX. The first is the standard format and is used for all non USB communication and incoming USB communication. The other two formats are used only by out going USB communication. One forms a list of endpoints, and the other forms lists of data to the individual endpoints. See *7.4.3 DMA Descriptor Format for USB.*

You can view the DMA list structure as a two dimensional list where the endpoint descriptors form one dimension, and by connecting a list of USB Sub-list descriptors to each endpoint descriptor, the USB Sublist forms the second dimension.

The same DMA list structure is used both when the USB controller is in Host mode, and when it is in Device mode. In Device mode, however, the endpoint list only contains one endpoint descriptor.

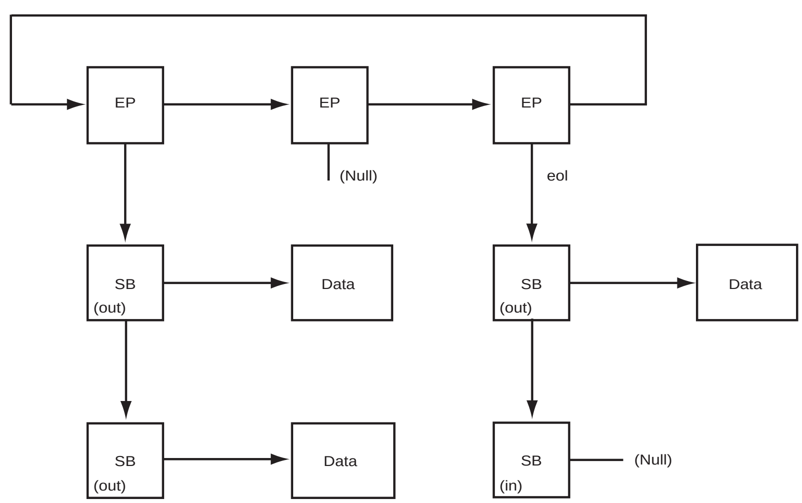


Figure 7-8 DMA List Structure for USB

There is no information in the descriptors telling which descriptor format a specific descriptor has. DMA interprets the descriptors based on how it is started, and on what internal state it is currently in.

7.4.3 DMA Descriptor Format for USB

The Standard Descriptor Format

The standard descriptor format in figure 7-9 below is similar to figure 7-6 with the following extensions of the 32-bit status field for channel 9 when connected to the USB interface.

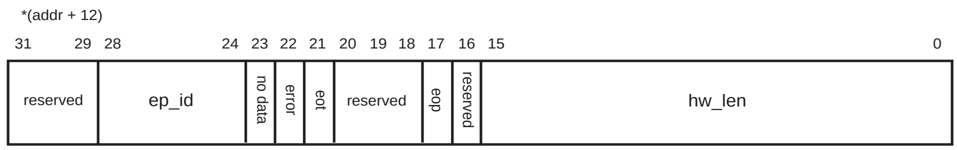


Figure 7-9 The DMA Descriptor Format for USB

This 32-bit field contains status information written by the DMA controller. Table 7-3 describes the contents of the standard DMA descriptor in more detail:

Bit	Name	Explanation
(addr + 12):		
31 - 29	reserved	(note 8)
28 - 24	ep_id	If eop == 1; ID of end point that is generating this data.
23	nodata	If eop == 1; There is no data in the buffer, and hw_len is not valid. Used when End Of USB Transfer was indicated by an empty USB packet.
22	error	If eop == 1; Received USB transaction has an error. Error status can be read in USB controller using ep_id as index. The corresponding endpoint is disabled to prevent status from being lost. (note 9)
21	eot	If eop == 1; End Of USB Transfer.
20 - 18	reserved	(note 8)
17	eop	Last descriptor in a received packet.
16	reserved	(note 8)
15-0	hw_len	If (eop == 1 && !nodata); Number of received bytes in buffer. (If all bits are 0, the length is 2^{16})

Table 7-3 DMA Descriptor contents for USB

Note 7: The 32-bit status field is only written if eop is set to one. For software to be able to know that the status field has been written or not, the status field of the descriptor must be cleared to all zeros before the DMA channel is started. The eop bit is written in the last descriptor of a packet.

Note 8: For compatibility with future versions of ETRAX processors, reserved fields must be set to 0 before starting DMA. When read, no assumption can be made about their value.

Note 9: For ISO IN endpoints, error can be the result of all USB: packet errors, no replies, or wrong packet sizes. No error is reported in the USB controller and the EP is not disabled.

USB EndPoint List Descriptor Format, EP

The DMA controller will only assume this format for sub channels of channel 8. Note that EndPoint descriptors must be 32-bit aligned.

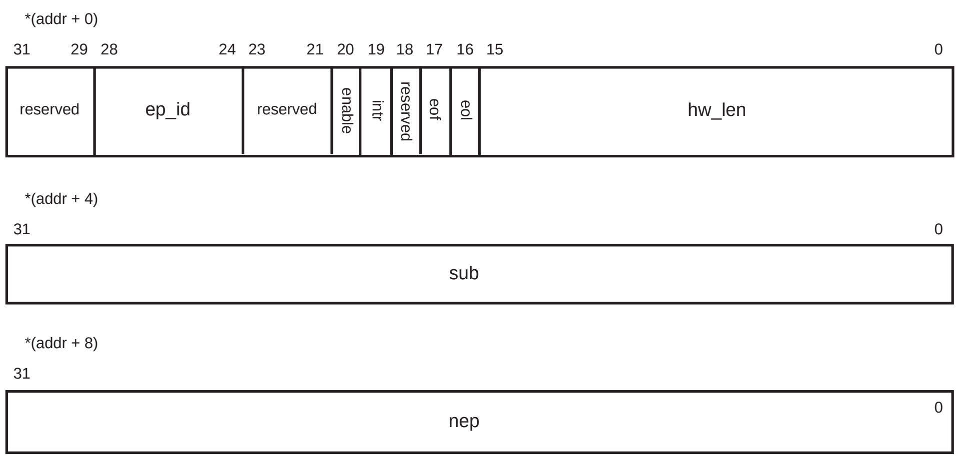


Figure 7-10 USB EndPoint List Descriptor Format, EP

The first 32-bit field of the descriptor gives a number of commands and an area hw_len for the DMA controller to save internal information. The second 32-bit field gives the address to the associate sub list. The last 32-bit field gives the address to the next EP descriptor in the EP list.

Table 7-4 describes the contents of the USB EndPoint list descriptor format, EP in more detail:

Bit	Name	Explanation
(addr + 0): (note 10)		
31 - 29	reserved	(note 11)
28 - 24	ep_id	Endpoint ID. Used by USB controller to associate a sublist with an endpoint. (note 12)
23 - 21	reserved	(note 11)
20	enable	Enable. 0: There is no sublist, advance to next endpoint. 1: There is a sublist, start processing the sublist. Note that software can only enable an endpoint, and that it has to be done according to the procedure in section *8.8.5 Managing SB Descriptor Lists in Host Mode* in Chapter *8 Universal Serial Bus*. To disable an endpoint, the EP descriptor must be removed from the endpoint list according to section *8.8.4 Managing EP Descriptor Lists in Host Mode*.
19	intr	Generate a dma8_subx_descr interrupt when advancing to next descriptor. (note 12)
18	reserved	(note 11).
17	eof	This is the last endpoint descriptor in a frame. Note that this bit will not generate any DMA interrupt. This field is not used in Device mode. (note 12)
16	eol	This is the last descriptor in the list. The list is assumed to be circular, and the nep field is used even if eol is true. (note 12)
15-0	hw_len	Counter used by the DMA controller to remember bytes left in current sublist data buffer. It should be cleared by software before the endpoint is enabled, and then ignored by software.
(addr + 4):		
31-0	sub	Pointer to first sublist descriptor in sublist. If enable == 0 sub is not used, there are no alignment restrictions on sublist descriptors.
(addr + 8):		
31-0	nep	Pointer to next endpoint descriptor in endpoint list. The two least significant bits in the nep field must be zero since it is a requirement that endpoint descriptors are 32-bit aligned. (note 10)

Table 7-4 USB EndPoint list descriptor contents

Note 10: EndPoint descriptors must be 32-bit aligned, i.e. addr<1:0> == 00.

Note 11: For compatibility with future versions of ETRAX processors, reserved fields must be set to 0 before starting DMA. When read, no assumption can be made about their value.

Note 12: The ep_id, intr, eof, and eol fields must not be changed when the enable field is equal to 1. Also, note that there is a special procedure to clear the enable field according to Table 7-4, and chapter *8.8.4 Managing EP Descriptor Lists in Host Mode.*

USB Sublist Descriptor Format, SB

The DMA controller will only assume this format for sub channels of channel 8.

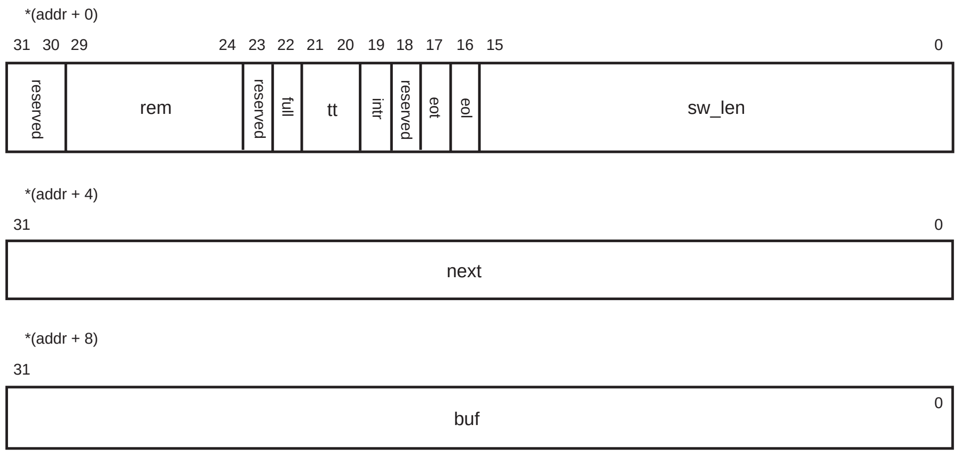


Figure 7-11 USB Sublist Descriptor Format, SB

The first 32-bit field of the descriptor gives a number of commands, and the length of the data buffer for outgoing transfers is given in sw_len. For Host mode IN transfers, sw_len gives the number of packets to be received by DMA channel 9. The second 32-bit field gives the address to the next descriptor in the linked list. The third 32-bit field gives the address to the data buffer buf for outgoing transfers. For Host mode IN transfers there is no data buffer appended, so buf should be set to NULL.

Table 7-5 describes the contents of the USB Sublist descriptor format, SB in more detail:

Bit	Name	Explanation
(addr + 0):		
31 - 30	reserved	(note 14)
29 - 24	rem	Expected number of bytes in the last packet of a Host mode IN transfer. Used to tell the USB controller the expected length of the last packet in a Host mode IN transfer. The expected number of packets in a Host mode IN transfer is the number of bytes in the data buffer at buf. This field is not used in Device mode.
23	reserved	(note 14).
22	full	If eot==1, full==1 tells the USB that an out transfer is a full length transfer. In the special case where the transfer-length is evenly divisible by the packetsize, this field prevents the USB sending an empty packet. In all other cases this field is don't care.
21 - 20	tt	USB Transfer Type: 00: zout (Zero length output transfer) 01: in (Input transfer) (not used in Device mode) 10: out (Output transfer) 11: setup (Setup transfer) (not used in Device mode)
19	intr	Generate a dma8_subx_descr interrupt when advancing to the next descriptor.
18	reserved	(note 14).
17	eot	This is the last descriptor in a USB transfer. Note that this bit will not generate any DMA interrupt.
16	eol	This is the last descriptor in the sublist. When eol is true (1), the next field is not used.
15-0	sw_len	Length in bytes of data buffer for outgoing transfers, and number of packets for Host mode IN transfers (If all bits are 0, the length is 2^{16}).
(addr + 4):		
31-0	next	Pointer to next descriptor in list (no alignment restrictions). If eol == 1 next is not used.
(addr + 8):		
31-0	buf	Pointer to first byte in the data buffer (no alignment restrictions). (note 13)

Table 7-5 USB Sublist descriptor contents

Note 13: For a Host mode IN transfer, DMA will not use the data referenced by buf.

Note 14: For compatibility with future versions of ETRAX processors, reserved fields must be set to 0 before starting DMA. When read, no assumption can be made about their value.

7.5 DMA Interrupt

There are two interrupts per channel: a descriptor interrupt, and an end-of-packet interrupt. Each channel has its own interrupt vector, and can be individually enabled or disabled. For a list of all interrupt vector numbers see chapter 17 *Interrupts*.

For output channels, the interrupts are generated if the intr bit (descriptor interrupt) or eop bit (end-of-packet interrupt) in the descriptor are set. The interrupts are generated after DMA has read all data from the associated data buffer. If the wait bit is set in the descriptor, the interrupt is delayed until the DMA FIFO is emptied by the connected I/O interface.

For input channels the descriptor interrupt is generated if the intr bit is set in the descriptor, and the end-of-packet interrupt is generated when the peripheral interface signals end-of-packet to the DMA controller. The interrupts are generated after DMA has written all the data to the associated data buffer.

Interrupts are cleared with a write to the R_DMA_CHx_CLR_INTR internal register. For a more detailed information regarding enable, disable, and read interrupts, refer to chapter 17 *Interrupts*, or chapter 18.14 *DMA Registers*.

7.6 DMA Transfer/Setup Examples

In the following examples, a pseudo code notation is used to access bit fields in internal registers: *REG.field*.

Here *REG* is the name of the register and *field* is the bit field in the register. Also, symbolic constants for values are used. See chapter 18.14 *DMA Registers* for a list of valid register, field, and constant names.

In addition, the following C structure is assumed for descriptors where a byte is an 8-bit unsigned integer, a word a 16-bit unsigned integer, and a dword a 32-bit unsigned integer:

```
struct descriptor
  {
    word sw_len;
    word ctrl;
    dword  next;
    dword  buf;
    word hw_len;
    byte status;
    byte fifo_len;
  }
```

For a information regarding USB to DMA transfer/setup examples, please refer to chapter 8.8 *Procedures*.

7.6.1 Initiate and Setup a DMA Transfer

To initiate and setup a DMA transfer you typically:

1. Connect the I/O interface to the desired DMA channel in R_GEN_CONFIG.
2. Initiate the I/O interface.
3. Reset the DMA channel by writing the reset-cmd to R_DMA_CHx_CMD.
4. Initiate the linked list.
5. Set R_DMA_CHx_FIRST to the first descriptor.
6. Start DMA by writing start-cmd to R_DMA_CHx_CMD.
7. Start the I/O interface.
8. Wait for the DMA transfer to end.

This is the typical sequence of steps to initiate and set up a DMA transfer. However, for some I/O interfaces the sequence has to be altered. See each interface's respective chapter for a description of possible alterations to the procedure above.

7.6.2 Reset DMA Channel

To reset a DMA channel and its FIFO, the reset-cmd is written to R_DMA_CHx_CMD. The reset takes a while (>100ns), and when it is done DMA clears R_DMA_CHx_CMD.

Pseudo C-macros, and code for a DMA reset:

```
#define reset_dma( n ) \
   R_DMA_CHn_CMD.cmd = reset

#define wait_reset_dma( n ) \
   while( R_DMA_CHn_CMD.cmd != hold )
```

Example:

```
/* Reset channel 0, 3, and 7. */
reset_dma( 0 );
reset_dma( 3 );
reset_dma( 7 );

/* Wait for reset do complete. */
wait_reset_dma( 0 );
wait_reset_dma( 3 );
wait_reset_dma( 7 );
```

7.6.3 Initiating Linked List

The following is a C-macro to initiate a descriptor, and how to use it to form a linked list. The next field is Don’t-Care at end of list.

```
#define descr( buf, len, next, flags ) \
  { len, flags, next, buf, 0, 0, 0 }
```

Example:

```
dma_descr olist[] = {
  descr( pattern0, sizeof pattern0, &olist[1], d_int | d_eop ),
  descr( pattern1, sizeof pattern1, &olist[2], 0 ),
  descr( pattern2, sizeof pattern2, 0xDC, d_eol )
};
```

7.6.4 Start a DMA Transfer

To start a DMA transfer R_DMA_CHx_FIRST and R_DMA_CHx_CMD have to be programmed. A sample C-macro for this is:

```
#define start_dma( n, desc ) \
  R_DMA_CHn_FIRST.first = (unsigned) desc; \
  R_DMA_CHn_CMD.cmd = start
```

Example:

```
start_dma( 0, olist ); /* Start channel 0 with olist. */
```

7.6.5 Restart a DMA Transfer

To append new data to a linked list the following steps should be taken. Note that the sequence of these steps is important:

1. Initiate the new list that shall be appended to the old list.
2. Set End-Of-List in the last descriptor of the new list.
3. Update the next field in the last descriptor of the old list.
4. Clear End-Of-List in the last descriptor of the old list.
5. Write the restart command to R_DMA_CHx_CMD.

The following code will work regardless of whether you give the restart command before or after DMA has finished the old list.

```
#define append_descr( n, old_last, new_first ) \
  old_last->next = new_first; \
  old_last->cmds =& ~d_eol; \
  R_DMA_CHn_CMD.cmd = restart
```

Example:

```
/* Initiate new list. */
dma_descr nlist[] = {
  descr( pattern3, sizeof pattern3, &nlist[1], d_int | d_eop ),
  descr( pattern4, sizeof pattern4, &nlist[2], 0 ),
  descr( pattern5, sizeof pattern5, 0xDC, d_eol )
};
append( 0, &olist[2], nlist );
```

7.6.6 Hold DMA Temporarily and Continue Later

To temporarily hold DMA the following two C *macros* could be used.

```
#define hold_dma( n ) \
  R_DMA_CHn_CMD.cmd = hold

#define continue_dma( n ) \
  R_DMA_CHn_CMD.cmd = continue
```

7.7 Memory to Memory DMA

DMA channels 6 and 7 can be set up for memory to memory DMA, where the FIFO buffers of channels 6 and 7 connect directly to each other. This is configured in R_GEN_CONFIG.

To make sure the last part of the memory to memory transfer is written out from the destination FIFO to external memory in the destination list, either the eop bit has to be set in the last descriptor of the source list, or the software has to manually flush the destination FIFO by writing to the R_DMA_SET_EOP register when DMA channel 7 has read all data from the source list and stopped.

7.8 External DMA Channels

The external DMA channels ext_dma0 and ext_dma1 offer a DMA-like interface between external memory and external I/O devices. The external DMA channels operate in a “pseudo DMA” fashion, which differs from an ordinary DMA operation in that the data from the memory or the I/O device pass through the FIFO buffer of an internal DMA channel (see figure 7-12). This construction has the advantage of the bus width and timing for the I/O device, being different from the bus width and timing for memory since the accesses are performed in separate bus cycles.

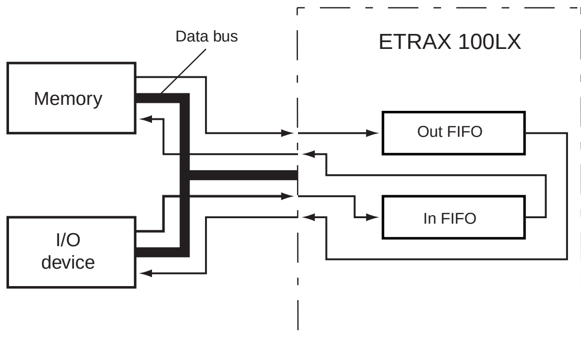


Figure 7-12 ETRAX 100LX External DMA Channels' Pseudo DMA Principle

Note 15: External DMA has the highest priority and might starve the other I/O interfaces such as Ethernet.

ext_dma0 and ext_dma1, which are available for external I/O, are bidirectional:

- ext_dma0 uses the internal DMA channels 4 and 5.
- ext_dma1 uses the internal DMA channels 6 and 7.

The data bus width of the DMA transfers can be configured to either 8, 16, or 32 bits. Each external DMA channel uses two control signals to enable a handshake procedure:

- DMA request (dreq0, dreq1)
- DMA acknowledge (dack0, dack1)

7.8.1 External DMA Configuration

The operational mode for external DMA channels ext_dma0 and ext_dma1 are configured in registers R_EXT_DMA_0_CMD and R_EXT_DMA_1_CMD. The following modes can be individually configured for each external channel:

- Width of the transfers (8, 16 or 32 bits)
- Direction of the transfers (in or out)
- Start and stop of the transfers
- Polarity of the request signal
- Polarity of the acknowledge signal
- Request/acknowledge mode (burst or handshake)
- Use of the transfer counter (on or off)

Before an external DMA channel is started, it needs to be connected to the corresponding internal DMA channels. This configuration is made in register R_GEN_CONFIG.

7.8.2 External DMA Address

External DMA bus cycles are identical to standard bus cycles, and will be directed to a constant, configurable address. This address may be used instead of, or in combination with the DMA acknowledge signal to decode the DMA cycles. The addresses for the two channels are configured in R_EXT_DMA_0_ADDR and R_EXT_DMA_1_ADDR. The two most significant bits of the address are always set to 10 (binary) so as to not access the internal cache memory or the DRAM (see figure 7-14).

31	30	29 … 2	1	0
1	0	External interface address	0	0

Figure 7-13 The Address Bus When Addressing an External Interface

7.8.3 Initialization

It is recommended that the external DMA channel is set up in the following sequence:

1 Connect the external DMA to its associated internal DMA channels. This is configured in R_GEN_CONFIG.

2 Configure the external DMA channel, but do not start it (configuration is made in: R_EXT_DMA_x_CMD and R_EXT_DMA_x_ADDR).

3 Reset and start the associated internal DMA channels.

4 Start the external DMA channel (run bit in R_EXT_DMA_x_CMD).

7.8.4 Request/Acknowledge Signaling

Each external DMA channel has a pair of handshaking signals, dreq and dack. Both can be configured to be active high or active low. There are two modes for the request/acknowledge signalling: *handshake mode* and *burst mode*.

Handshake Mode

In handshake mode, a 4-phase handshaking scheme is used:

1 The I/O device sets dreq active.

2 The ETRAX 100LX sets dack active and performs a read or write operation to the I/O device (i.e. an external DMA bus cycle).

3 The I/O device negates dreq. This can be done immediately after dack gets active, and does not have to wait for the bus cycle to complete.

4 The ETRAX 100LX negates dack. The ETRAX 100LX will never negate dack before the external DMA bus cycle is completed.

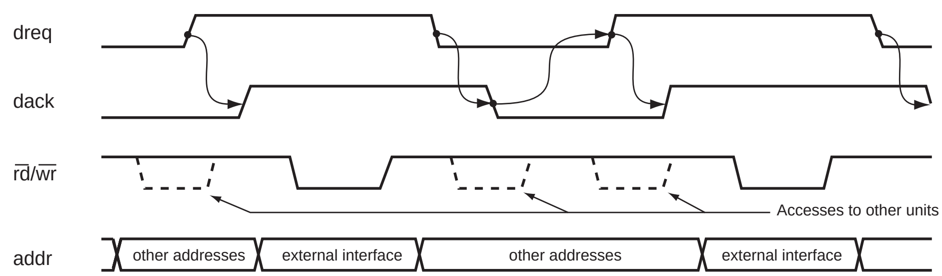


Figure 7-14 Timing Diagram for Handshaking Mode

Burst Mode

In burst mode, the ETRAX 100LX will always release the dack signal immediately after the completion of the external DMA bus cycle, and will continue to issue new external DMA bus cycles as long as the I/O device keeps the dreq active. Each cycle will be accompanied by a dack pulse. If the I/O device wants to stop or pause the transfers, it has to negate the dreq immediately after receiving the dack for the last cycle.

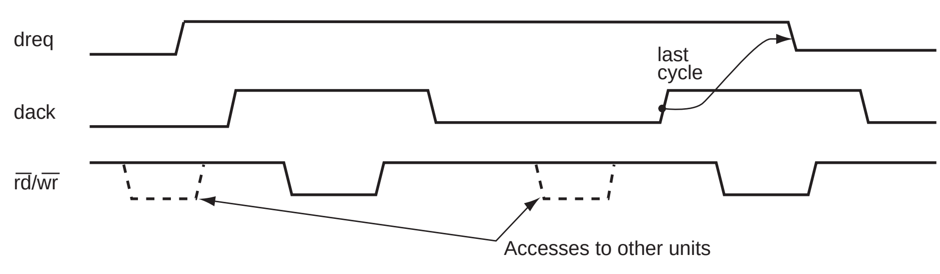


Figure 7-15 Timing Diagram for Burst Mode

7.8.5 Start and Stop of the Transfers

After initializing the external DMA (see section *7.8.3 Initialization*), the external DMA interface is started by setting the run bit in the R_EXT_DMA_x_CMD register. When the external DMA channel is running, the external I/O device can start and stop the transfers by activating and deactivating the dreq signal.

Except for dreq going inactive, the transfers could stop for one of the following reasons:

- The external DMA channel is stopped by the software, by setting the run bit in R_EXT_DMA_x_CMD to stop.
- The transfer counter (see below) is in use and has expired.
- The descriptor list of the associated internal DMA channel contained an eop flag (only in the case of transfers from ETRAX 100LX to the I/O device).
- The data buffers of the associated internal DMA channel are exhausted.

If one of the first two cases occurs during input from the external I/O to the ETRAX 100LX, an eop status bit will be set in the corresponding DMA descriptor in the descriptor list for the associated internal DMA channel, and the internal DMA channel will advance to the next descriptor.

7.8.6 Transfer Counter

The number of transfers performed by the external DMA channel can be controlled by a transfer counter (tfr_count field in register R_EXT_DMA_x_CMD). The transfer counter is configured with the desired number of transfers (where a transfer is either 8, 16, or 32 bits), and will be decremented by one for each transfer. When the counter reaches zero, it will stop the transfers and clear the run bit in R_EXT_DMA_x_CMD. The current values of the run bit and the transfer counter can be read from the register R_EXT_DMA_x_STAT.

7.8.7 External DMA Interrupts

Each external DMA channel will generate an interrupt, ext_dma0 and ext_dma1 respectively, whenever the run bit in R_EXT_DMA_x_CMD is cleared. When not waiting for an external DMA transfer to complete, the interrupt should be masked off in R_IRQ_MASK0_SET. For further information about interrupts refer to chapter 17 *Interrupts*.

8 UNIVERSAL SERIAL BUS

ETRAX 100LX includes an on-chip Universal Serial Bus (USB) interface that complies with the Universal Serial Bus Revision 1.1 specification.

The interface is equipped with two USB ports designated p1 and p2 respectively, the inputs and outputs of which are multiplexed on to the same pins as other interface applications (see chapter 19 *Electrical Information*). The characteristics and operational principles of both ports are similar. The electrical interface is compatible with a Philips model PDIUSBP11A transceiver (or equivalent), as illustrated below.

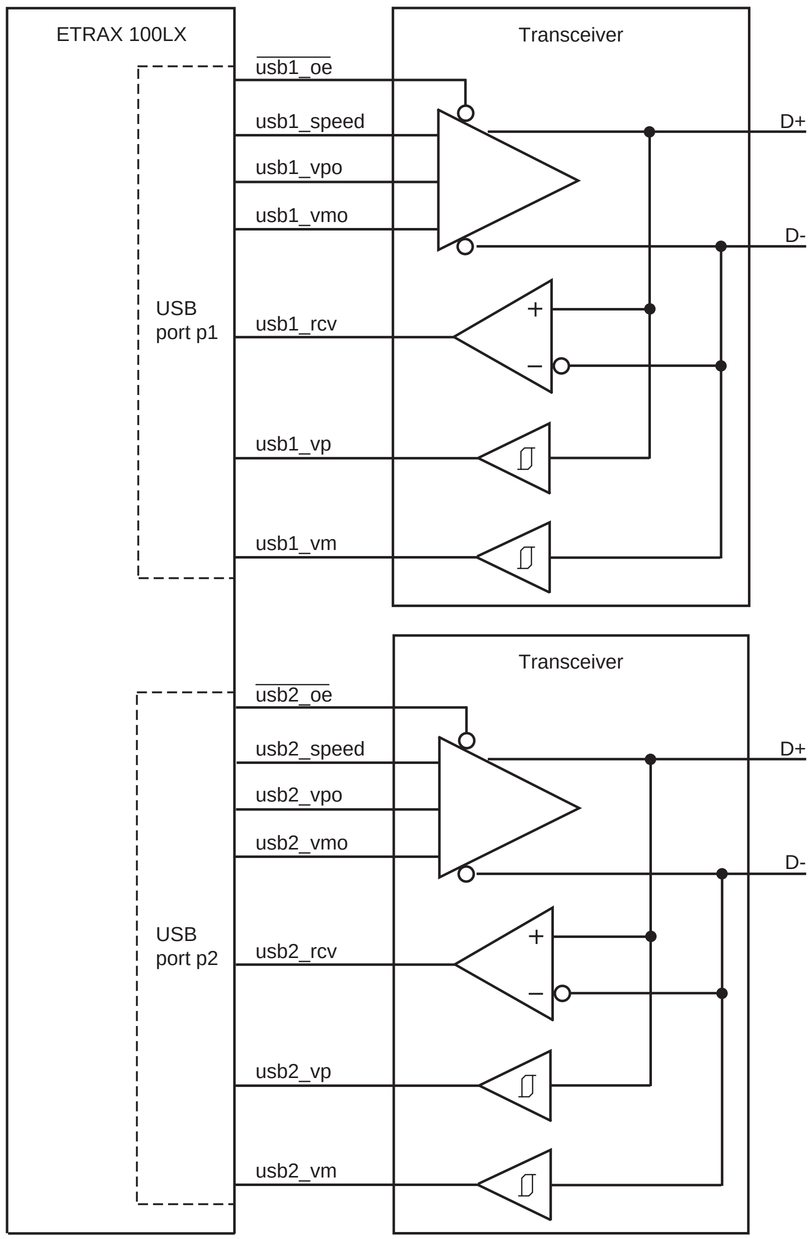


Figure 8-1 USB Transceiver Interface

8.1 Principle of Operation

8.1.1 Basic Architecture of the USB Interface

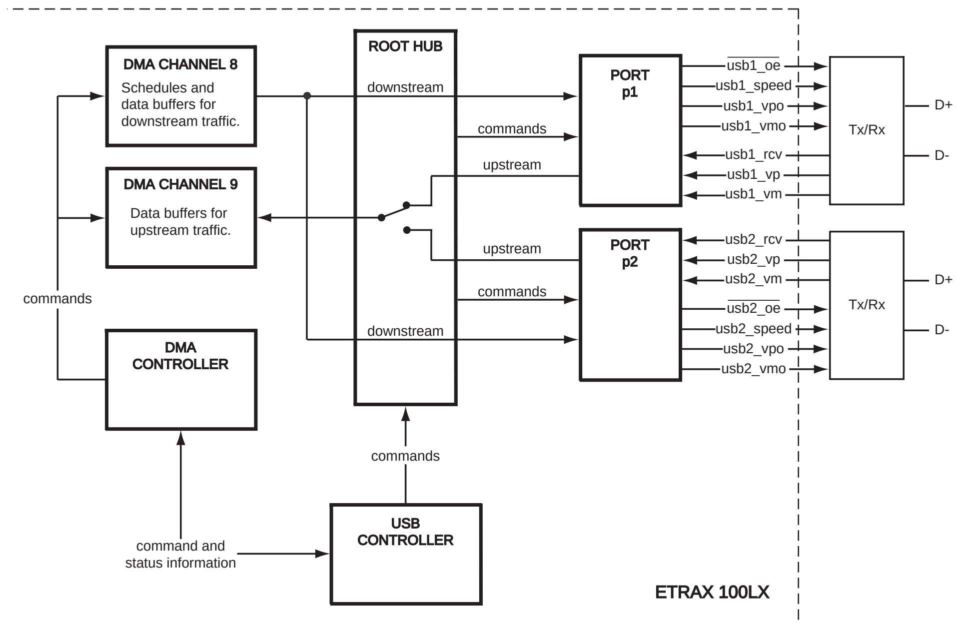


Figure 8-2 Basic USB Architecture

The USB is comprised of a root hub with ports p1 and p2 and the USB controller. The ports are duplex, and each port is connected to a dedicated transceiver (via I/O multiplexing not shown on the diagram). The root hub in the data stream controls the two USB ports and is itself managed by the USB controller, which handles frame control and timing, transaction protocol, port management and error recovery. In device mode, the root hub is only used as a port controller.

The USB interface operates in cooperation with the Direct Memory Access (DMA) controller, which manages all traffic schedules and communicates with the USB via two DMA channels. DMA channel 8 handles the traffic schedules and data buffers for downstream (OUT) traffic: DMA channel 9 handles the data buffers for upstream (IN) traffic. Please refer to chapter 7 *DMA* for detailed information on DMA.

8.1.2 Modes of Operation of the USB Interface

The USB interface can be configured to operate in one of two modes:

Host mode - in which ETRAX 100LX acts as the host for devices connected to the USB. In this mode, the USB interface can use either or both ports.

Device mode - in which ETRAX 100LX acts as a device communicating with a host elsewhere. Either of the two USB ports can be used in this mode, but not both simultaneously.

8.2 Operational States of the USB Controller

The USB controller has six operational states which are illustrated below.

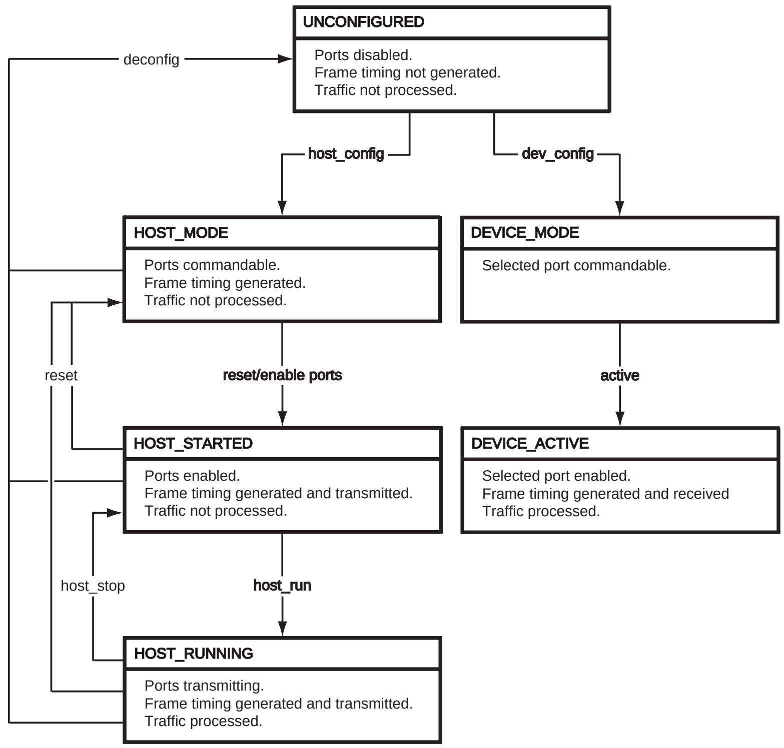


Figure 8-3 Operational States of the USB Controller

The UNCONFIGURED state prevails immediately after the execution of a **deconfig** command or system reset. The USB controller is stopped, the ports are disabled, frame timing is not generated and no traffic is processed.

HOST_MODE prevails in response to a **host_config** command while in the UNCONFIGURED state, or a **reset** command while in the HOST_STARTED or HOST_RUNNING state. In HOST_MODE the USB controller is prepared to act as a host. Commands can be issued to the ports and frame timing is generated, but no traffic is processed.

The HOST_STARTED state prevails when one or both ports have been enabled in HOST_MODE, or in response to a **host_stop** command in the HOST_RUNNING state. In the HOST_STARTED state, frame timing is generated and transmitted but traffic is not processed.

The HOST_RUNNING state prevails in response to a **host_run** command in the HOST_STARTED state. This is the fully operational state of the USB controller in Host mode, with the configured port(s) handling processed traffic.

DEVICE_MODE prevails in response to a **dev_config** command while in the UNCONFIGURED state. The USB controller is prepared to act as a device and commands can be issued to the selected port.

The DEVICE_ACTIVE state prevails in response to an **active** command in the DEVICE_MODE. This is the fully operational state of the USB controller in Device mode, with frame timing generated and the configured port handling traffic.

Please refer to section 8.4.1 *USB Controller Commands in Host Mode* for information on the USB controller commands.

8.3 USB Registers

8.3.1 Register Access Timing

There is a short delay between writing to a mode register and its effect on the hardware. Consequently, a **NOP** (no operation) instruction must be inserted between a write operation and a succeeding read operation to the same or affected register in order to read the updated value. This needs to be considered, for example, when accessing the EP Table.

8.3.2 USB Mode Registers

The USB interface is served by a set of dedicated mode registers. Table 8-1 below introduces these registers and summarizes their functions. For more detailed information on the USB registers, please refer to chapter 18.16 *Universal Serial Bus Interface Control Registers*.

Register	Function
R_USB_COMMAND	A byte-wide, read/write register that controls the USB commands in Host mode. Its functions include port selection, host and device configuration/deconfiguration, reset, run and stop commands.
R_USB_COMMAND_DEV	A byte-wide, read/write register that controls the USB commands in Device mode. The commands are similar to those used in Host mode.
R_USB_STATUS	A byte-wide, read-only register containing controller status information. The fields show whether the controller is busy, a Host or Device mode indicator, and whether Host mode is started and running.
R_USB_IRQ_MASK_SET	A 16-bit wide, write-only register in which ten control bits are used to enable or mask separate USB interrupts in Host mode. They are: isochronous end of frame (**iso_eof**); interrupt end of frame (**intr_eof**); isochronous end of transfer (**iso_eot**); interrupt end of transfer (**intr_eot**); control end of transfer (**ctl_eot**); bulk end of transfer (**bulk_eot**); endpoint attention (**epid_attn**); start of frame (**sof**); port status (**port_status**); controller status (**ctl_status**).
R_USB_IRQ_MASK_SET_DEV	A 16-bit wide, write-only register in which nine control bits are used to enable or mask separate USB interrupts in Device mode. They are: end of transfer/transaction for OUT (**out_eot**); endpoint3 end of transfer (**ep3_in_eot**); endpoint2 end of transfer (**ep2_in_eot**); endpoint1 end of transfer (**ep1_in_eot**); endpoint0 end of transfer (**ep0_in_eot**); endpoint attention (**epid_attn**); start of frame (**sof**); port status (**port_status**); controller status (**ctl_status**).

Table 8-1 USB Mode registers

R_USB_IRQ_MASK_READ	A 16-bit wide, read-only register that shows the status of the USB interrupts after individual bit masking in Host mode. Its contents are controlled by R_USB_IRQ_MASK_SET and R_USB_IRQ_MASK_CLR.
R_USB_IRQ_MASK_READ_DEV	A 16-bit wide, read-only register that shows the status of the USB interrupts after individual bit masking in Device mode. Its contents are controlled by R_USB_IRQ_MASK_SET_DEV and R_USB_IRQ_MASK_CLR_DEV.
R_USB_IRQ_MASK_CLR	A 16-bit wide, write-only register with ten control bits that are used to clear the USB interrupt mask bits in Host mode.
R_USB_IRQ_MASK_CLR_DEV	A 16-bit wide, write-only register with nine control bits that are used to clear the USB interrupt mask bits in Device mode.
R_USB_IRQ_READ	A 16-bit wide, read-only register containing ten pending USB interrupt bits. It shows the status of the USB interrupts prior to individual bit masking in Host mode.
R_USB_IRQ_READ_DEV	A 16-bit wide, read-only register containing nine pending USB interrupt bits. It shows the status of the USB interrupts prior to individual bit masking in Device mode.
R_USB_FM_NUMBER	A 32-bit wide, read/write register that reads the number of the current USB frame in Host mode. The register is cleared when the USB controller is reset. Reading this register clears the sof interrupt condition.
R_USB_FM_NUMBER_DEV	A 32-bit wide read/write register that contains the current frame number and Host/Device frame synchronization information in USB Device mode. The lower 11 bits represent the number of the current frame. The 8 msb bits read the time difference between the Host start-of-frame and the Device frame timer. The register is cleared when the USB controller is reset. Reading this register clears the sof interrupt condition.
R_USB_FM_INTERVAL	A 16-bit wide, read/write register containing a 14 bit value that defines the bit time interval in a frame (the distance between two start-of-frames). The frame timer decrements from this value to zero. The value is reloaded into register R_USB_FM_REMAINING at each start-of-frame (SOF). The value in this register is the frame length minus 1.
R_USB_FM_REMAINING	A 16-bit wide, read-only register that holds the remaining number of bit times in the current frame. The lower 14 bits represent this value: the two msb are not used.
R_USB_FM_PSTART	A 16-bit wide, read/write register that holds the periodic start point. The lower 14 bits represent this value: the two msb are not used. The value of the periodic start point is compared with a value counted downwards.
R_USB_RH_STATUS	A byte-wide, read-only register that contains root hub status information about the USB ports. Two 2-bit fields hold the bus states of ports p1 and p2 as sampled at the EOF2 time in the frame. A third 2-bit field represents the general condition of the USB interface.
R_USB_RH_PORT_STATUS_1	A 16-bit wide, read-only register containing status information about USB port p1. It is compatible with the **wPortStatus** field of the **get_status** command to USB hubs. The parameters that are read are **speed**, **reset**, **suspended**, **enabled** and **connected**. Two fields are not implemented in hardware and must therefore be handled in software. They are **over_current** (bit 3) and **port_power** (bit 8). When this register and R_USB_RH_PORT_STATUS_2 are read, the root hub status change interrupt condition is cleared. The register must be read as an entire word (16-bits).
R_USB_RH_PORT_STATUS_2	A 16-bit wide, read-only register containing the same information as R_USB_RH_PORT_STATUS_1, but for USB port p2.

Table 8-1 USB Mode registers

R_USB_EPT_INDEX	A byte wide, read/write register in which the five lsb contain the index of the endpoint lookup table to be used when reading and writing via register R_USB_EPT_DATA. The endpoint lookup table contains 32 endpoint entries, each pointing at an endpoint on a USB device. The table is indexed by the endpoint ID (**ep_id**) number in the USB DMA descriptors.
R_USB_EPT_DATA	A 32-bit wide, read/write register. It is the general endpoint table data register for normal (non-isochronous) transfers in Host mode. Bit 18 of the register is unused (reserved). The other fields contain: **valid** - validity of the table entry; **hold** - software exclusion bit; **error_count_in** - error counter for incoming transactions; **t_in** - toggle bit for incoming transactions; **low_speed** - endpoint low speed marker: **port** - indicates the upstream device traffic port; **error_code** - error type indicator; **t_out** - toggle bit for outgoing transactions; **error_count_out** - error counter for outgoing transactions; **max_len** - maximum length of non-isochronous data packets; **ep** - endpoint number; **dev** - configured device address.
R_USB_EPT_DATA_ISO	A 32-bit wide, read/write register. It is the general endpoint table data register for isochronous transfers in Host mode. The register is similar to R_USB_EPT_DATA.
R_USB_EPT_DATA_DEV	A 32-bit wide, read/write register. It is the general endpoint table for Device mode transfers. The register is similar to R_USB_EPT_DATA.
R_USB_EPID_ATTN	A 32-bit wide, read-only register. It contains a value representing an EP table entry that merits software attention. Reading this register clears the **epid_attn** interrupt condition.
R_USB_PORT1_DISABLE	A byte-wide register in which one bit is used to disable USB port p1.
R_USB_PORT2_DISABLE	A byte-wide register in which one bit is used to disable USB port p2.

Table 8-1 USB Mode registers

8.4 USB Host mode

In Host mode the USB interface manages frame timing, transaction protocol, port management and error recovery. The four transfer types stipulated in the USB specification are supported, namely: control (CTRL), bulk (BULK), interrupt (INTR) and isochronous (ISO).

Either or both of the USB ports can be used. Selection of the port(s) to be used is made by asserting fields **usb1** (bit 29) and **usb2** (bit 30) in general configuration register R_GEN_CONFIG. The act of port selection also enables the USB interface. Prior to port selection, the interface and its registers are entirely inoperative.

The root hub controls the reset, disable, suspend, resume, receive and transmit activities of the USB ports, even if only one port is configured for use. It is commanded by the **port_sel** (bits 7 and 6) and **port_cmd** (bits 5 and 4) fields in register R_USB_COMMAND. The state of the ports is read in registers R_USB_RH_PORT_STATUS_1 and R_USB_RH_PORT_STATUS_2. The status of the root hub is read in register R_USB_RH_STATUS.

8.4.1 USB Controller Commands in Host Mode

Register R_USB_COMMAND is used to control the USB interface in Host mode. This register contains fields for commanding the root hub and the USB controller. All these fields must be written in one operation.

Each time a write operation to register R_USB_COMMAND is performed, a command interpretation is triggered. This sets the **busy** field (bit 7) in register R_USB_STATUS. When the USB controller has executed the command, the **busy** field is cleared. The current state of the USB interface is read in register R_USB_STATUS.

The commands to the USB controller in Host mode are:

nop - (no operation)
One or both ports can be commanded without issuing any commands to the USB controller. The **nop** command is issued by setting field **ctrl_cmd** (bits 2 to 0) in register R_USB_COMMAND to the value 0x0.

reset
When the status of the USB controller is HOST_STARTED or HOST_RUNNING, this command resets the USB controller. It overrides any port commands by disabling either or both configured ports. The state of the USB controller changes to HOST_MODE, which is functionally equivalent to executing the **deconfig** and **host_config** commands in succession. The **reset** command is issued by setting the **ctrl_cmd** field in R_USB_COMMAND to the value 0x1.

deconfig
This is an emergency stop command that immediately deconfigures the entire USB interface, returning it to the condition that immediately succeeds a reset. The state of the USB controller changes to UNCONFIGURED. The **deconfig** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value 0x2.

host_config
With the USB controller in the UNCONFIGURED state, this command configures it as a host controller. Any command to a port is overridden by a configuration request to the port to adopt Host mode. The state of the USB controller changes to HOST_MODE and, as soon as a port is reset and enabled, the status changes to HOST_STARTED. The **host_config** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value 0x3.

dev_config
Please refer to section 8.5 *USB Data Structures in Host Mode*, Device mode. The **dev_config** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value 0x4.

host_nop
The **host_nop (no operation)** command is issued by setting the **ctrl_cmd** field in R_USB_COMMAND to the value 0x5.

host_run
When the host is started, this command starts the processing of USB traffic, and the status of the controller changes to HOST_RUNNING. It is not necessary to set up the various data structures for USB traffic before issuing the **host_run** command, but no traffic processing will occur until this has been done. The **host_run** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value 0x6.

host_stop
If the controller status is HOST_RUNNING, this command stops all traffic processing and changes the status of the controller to HOST_STARTED. The **host_stop** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value 0x7.

Most commands to the USB controller are effective in certain states only. The table below summarizes the relationship between the commands and the state of the USB controller.

	ctrl_cmd							
Status	**nop**	**reset**	**deconfig**	**host_config**	**dev_config**	**host_nop**	**host_run**	**host_stop**
UNCONFIGURED	nop	nop	OVR	OVR	OVR	nop	nop	nop
HOST_MODE	nop	nop	OVR	nop	nop	nop	nop	nop
HOST_STARTED	nop	OVR	OVR	nop	nop	nop	OK	nop
HOST_RUNNING	nop	OVR	OVR	nop	nop	nop	nop	OK

In the table above:

OVR indicates that the **ctrl_cmd** field in register R_USB_COMMAND overrides the **port_sel** and **port_cmd** fields.

OK and **nop** indicate that the **port_sel** and **port_cmd** commands are executed.

Note 1: If the USB controller is in the UNCONFIGURED state, then the port_cmd command cannot be executed.

If a command is written to register R_USB_COMMAND while the busy bit (3) in register R_USB_COMMAND is set, the result is unpredictable.

8.4.2 USB Port (Root Hub) Commands in Host Mode

The root hub operates in conjunction with the USB controller. Commands to the root hub are given in the **port_cmd** field of register R_USB_COMMAND. Any combination of the **ctrl_cmd** and **port_cmd** values can be used, but some are inappropriate. For instance do not command the USB controller to assume the HOST_RUNNING state whilst simultaneously disabling both of the USB ports. Instead, set the **ctrl_cmd** field of register R_USB_COMMAND to 0 (**nop**), and then use the **port_sel** and **port_cmd** fields to command the root hub.

The commands to the USB ports in Host mode are issued to the port(s) selected in the **port_sel** field (bits 7 to 6) in register R_USB_COMMAND as follows:

reset
The selected port is reset and enabled. A full port initialization of the port is performed and a bus reset is signalled according to the USB protocol. This occupies approximately 3 ms, depending upon the point in the frame cycle at which the command is issued. The **reset** command is issued by setting the **port_cmd** field in register R_USB_COMMAND to the value 0x0.

disable
The selected port is disabled. The **disable** command is issued by setting the **port_cmd** field in register R_USB_COMMAND to the value 0x1.

suspend
The selected port is placed in a suspended state. The port will not forward outbound traffic but will detect remote wakeup signalling or disconnects on the bus. The **suspend** command is issued by setting the **port_cmd** field in register R_USB_COMMAND to the value 0x2.

resume
The selected port is forced to resume operation. It starts a resume signalling sequence on the USB, after which the forwarding of USB traffic commences. The **resume** command is issued by setting the **port_cmd** field in register R_USB_COMMAND to the value 0x3.

The root hub handles most of the signalling on the bus. Functions such as connect/ disconnect detection, reset signalling, suspend/resume and speed detection are automatic and accord with the USB specification. Some device attachment and configuration timeouts must be handled in software.

8.5 USB Data Structures in Host Mode

8.5.1 Transfer Frames

A USB transfer frame has the basic structure shown below:

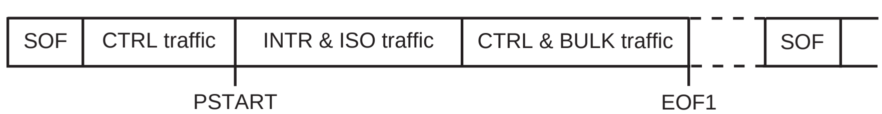


Figure 8-4 Basic Structure of USB Transfer Frame in Host Mode

The significant points in the transfer frame are the start-of-frame (SOF), periodic start (PSTART) and end-of-frame (EOF) events.

Start of Frame

The SOF point signifies the start of a transfer frame. The period from the beginning of one SOF to the next is the bit time interval of the frame. This is loaded into the **value** field (bits 13:0) of register R_USB_FM_REMAINING at the start of each frame. The frame timer decrements from the bit time interval to zero, at which point the next SOF occurs.

The bit time interval is a 14-bit value contained in the **fixed** and **adj** (adjustable) fields of register R_USB_FM_INTERVAL. The default value is 0x2EDF. The **fixed** field (bits 13:6) contains the upper 8-bits of the frame interval and, as the field name suggests, they are read-only.

The **adj** field (bits 5:0) contains the lower 6-bits of the frame interval. These bits can be used to adjust the bit time interval if necessary. Revision 1.1 of the USB specification requires that the interval must not deviate from the nominal (12000 bit times), by more than 15 bit times. Moreover the software must not adjust the frame interval by more than one bit time over six frames, and only one bit time is permitted in each adjustment.

Note 2: The value in register R_USB_FM_INTERVAL is actually the frame length minus 1.

Periodic Start Point

PSTART is where the transfer of interrupt and isochronous traffic can commence. Prior to the PSTART mark, only control transfers are performed. The position of PSTART is set by the 14-bits of the **value** field (13:0) in register R_USB_FM_PSTART.

To determine when to start sending periodic traffic, the USB controller compares R_USB_FM_PSTART with R_USB_FM_REMAINING, which is counted downwards. The USB driver should set R_USB_FM_PSTART to 0x2A30 (Periodic traffic starts 10% after sof).

More control traffic is permitted when all interrupt and isochronous transactions in the current frame are finished. When all remaining control traffic is finished, then bulk traffic transactions are performed.

End of Frame

At the EOF point, all traffic for the current frame is stopped and the entire procedure is repeated in the next frame. The register R_USB_FM_REMAINING shows the number of bit times remaining in the current frame. Register R_USB_FM_NUMBER gives the 32-bit frame number of the current frame.

Note 3: A 32-bit frame number is used in Host mode only. The lower 11 bits of register R_USB_FM_NUMBER are sent at the SOF.

Transfers and Toggle Bits

The USB interface does not set or clear the toggle bits at the end of a transfer. In fact the toggle bits are initialized only during the setup transaction in a CTRL transfer.

This feature can be used to concatenate multiple short transfers (as presented to the USB interface), into one long transfer (as seen by the device). It also means that the software may have to preset the toggle bits between transfers. This affects the insertion of transfers in the SB descriptor lists.

Maximum Transfer Length

Revision 1.1 of the USB specification does not specify a maximum transfer length. The USB interface is designed to construct longer transfers by concatenation.

Please refer to Universal Serial Bus Revision 1.1 specification for more detailed information about toggle bits, transfers and transfer length.

8.5.2 DMA Descriptors

The DMA generates the traffic for the USB interface, using linked lists of descriptors to track the USB traffic data and schedules. For further information regarding DMA refer to chapter 7 *DMA*.

DMA channel 8 is used for the traffic schedule and outgoing traffic data. DMA channel 9 is used for incoming traffic data. All incoming traffic terminates in the same buffer list in DMA channel 9.

Each traffic type has its own data structure, and the DMA divides channel 8 into four sub-channels (8.0 to 8.3), one for each traffic type, each of which is a linked list of linked lists. Sub-channel 8.0 is used for bulk traffic, sub-channel 8.1 for control traffic, sub-channel 8.2 for interrupts and sub-channel 8.3 for isochronous traffic.

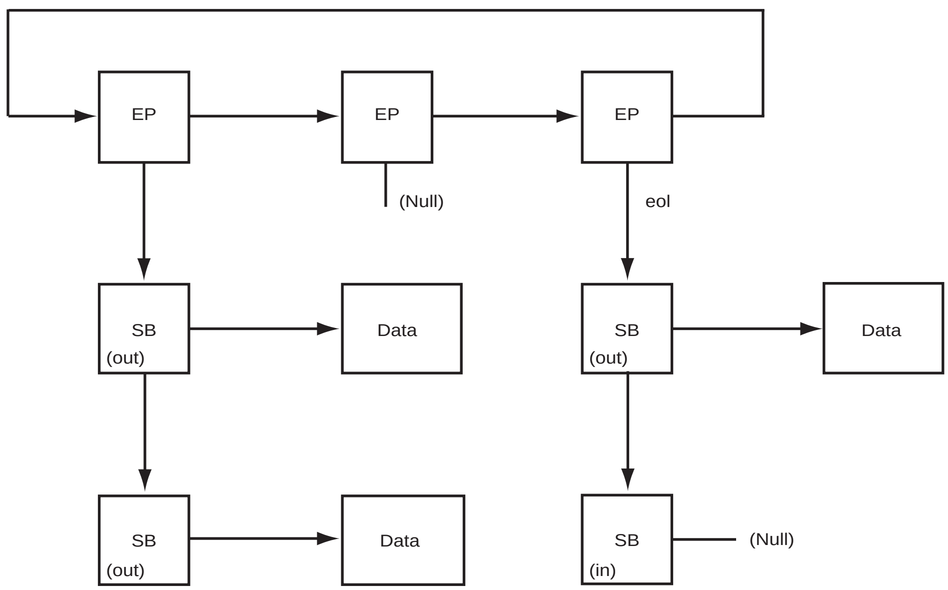


Figure 8-5 DMA List Structure for USB

In the upper dimension, the endpoint (EP) descriptors are aligned. From each EP descriptor, a number of sublist (SB) descriptors descend to form a transfer list for that endpoint. The EP list has exactly one descriptor with the **eol** (end-of-list) flag set.

An EP descriptor can be enabled only by software, but the hardware and the software can both disable an endpoint. When the hardware disables an EP descriptor, the software is notified by means of the **epid_attn** interrupt (See section 8.5.4 *Host Mode Interrupts*).

The SB descriptors describe the transfer to be performed. There are four types of transfer:

ZOUT - a special transfer used by the control, interrupt and isochronous traffic types, but not for bulk traffic.

SETUP - this transfer type is used for control transfers only.

IN and **OUT -** these transfer types are used in all four types of traffic. The SB descriptor for the OUT transfer has an associated buffer for the outgoing data, whereas the IN transfer data is received by DMA channel 9.

DMA Descriptors for IN Transfers

The format of a DMA descriptor for an IN transfer, used by DMA channel 9, is shown below. It is recommended that all DMA descriptors for USB are 32-bit aligned due to performance.

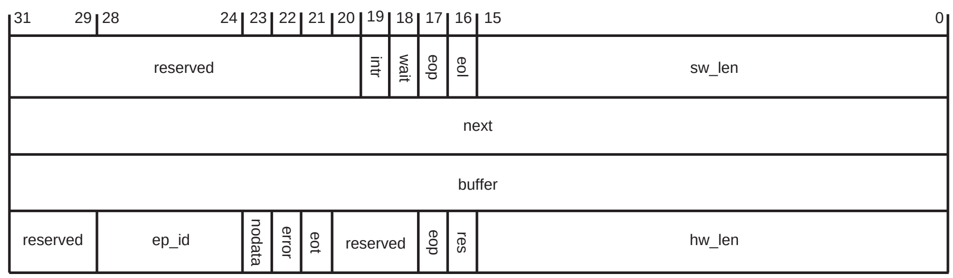


Figure 8-6 Format of DMA Descriptor

For a detailed description of the DMA descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB.*

Endpoint Descriptors

The format of an EP descriptor, used by DMA channel 8, is shown below. The **ep_id** field reports the EP identifier with which this buffer is associated, and is used to demultiplex the input stream into the corresponding endpoint pipe streams. All EP descriptors must be 32-bit aligned.

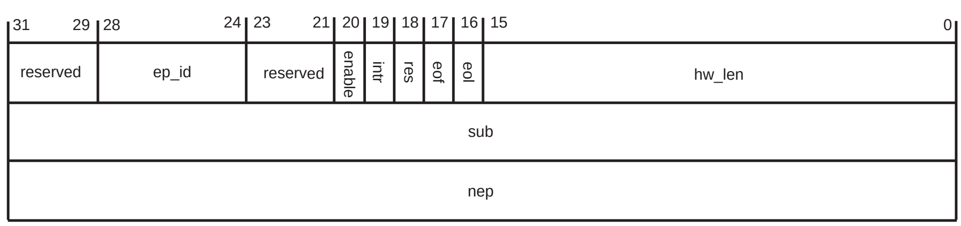


Figure 8-7 Format of an EP Descriptor

For a detailed description of the EP descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB.*

Sublist Descriptors

The format of an SB descriptor, used by DMA channel 8, is shown below.

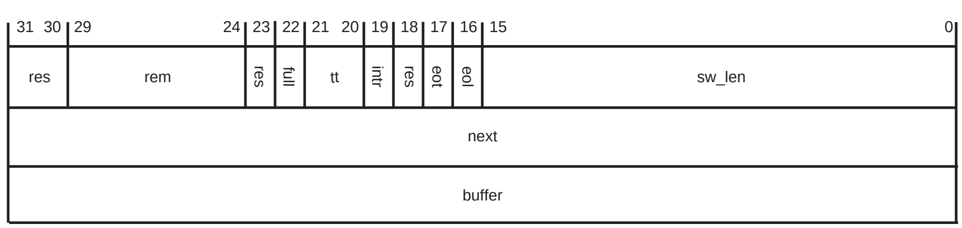


Figure 8-8 Format of SB Descriptor

In the special case where the transfer length is evenly divisible by the packet length, the full field is used to prevent the USB controller from sending an empty packet at the end of the transfer.

The rem field is used by IN transfers to count the bytes in the last packet. If the last packet is expected to be full, then the rem field is set to zero. For a detailed description of the SB descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB*.

SB Descriptors for Bulk Traffic

The transfers in the sublists can be IN or OUT. Each transfer can be constructed from one or more SB descriptors, but the last descriptor in a transfer must have the **eot** flag set. All descriptors in a single transfer must be of the same transfer type with the last descriptor in a sublist having the **eol** flag set.

In the examples that follow, the maximum packet size is set in the EP table (See section 8.5.3 *Endpoint Table in Host Mode*).

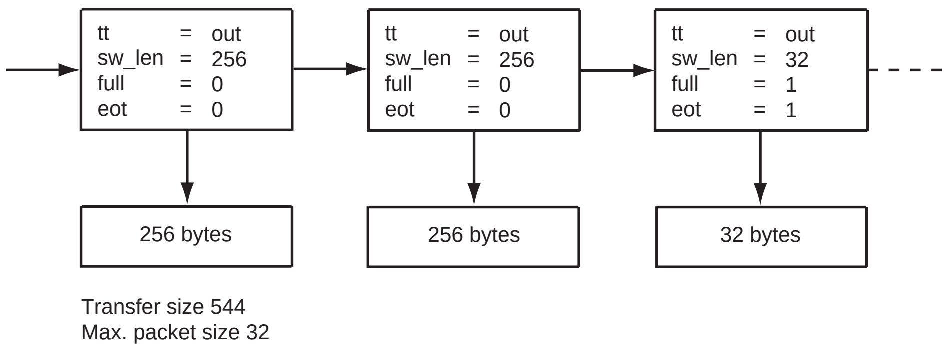


Figure 8-9 Example 1 of an OUT Transfer

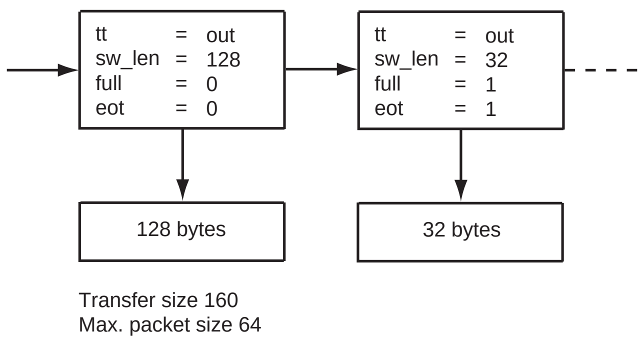


Figure 8-10 Example 2 of an OUT transfer

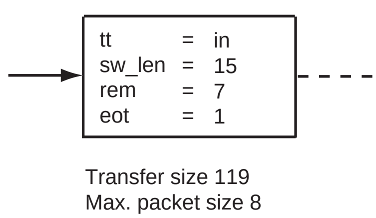


Figure 8-11 Example of an IN transfer

The calculations are:

sw_len = size ? (size - 1) / max packet size + 1 : 0;
rem = size % max packet size;

Note 4: For IN transfers, the sw_len field counts in packets.

SB Descriptors for Control (CTRL) Traffic

The format of a control transfer is more complicated than that of other traffic types. The USB specification requires that a control transfer must have either two or three phases: *setup*, *data* (optional), and *status* - and each phase uses one SB descriptor. For example, to build a control write it is necessary to have three SB descriptors: one SETUP, one OUT and one IN.

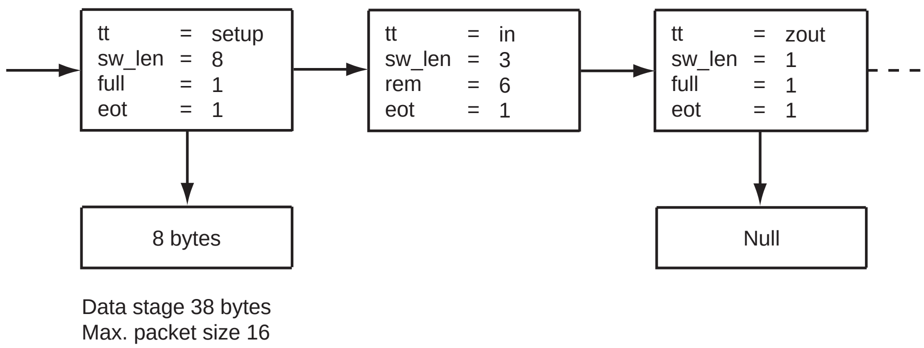


Figure 8-12 Example of a Control Read Transfer

Note 5: According to the Universal Serial Bus Revision 1.1 specification, the setup transaction must be exactly 8 bytes long.

The ZOUT descriptor buffer pointer should be set to NULL.

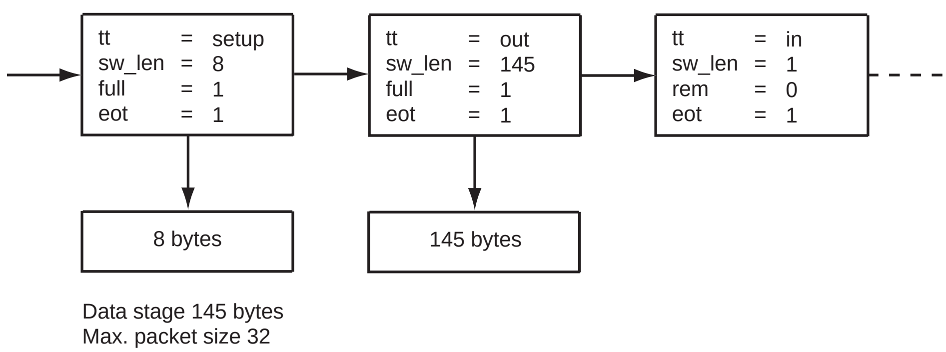


Figure 8-13 Example of a Control Write Transfer

Note 6: The **sw_len** field in the last descriptor has to be exactly 1. This is due to a requirement of the USB specification that a host must be prepared to receive data (from a bogus device) in the status phase.

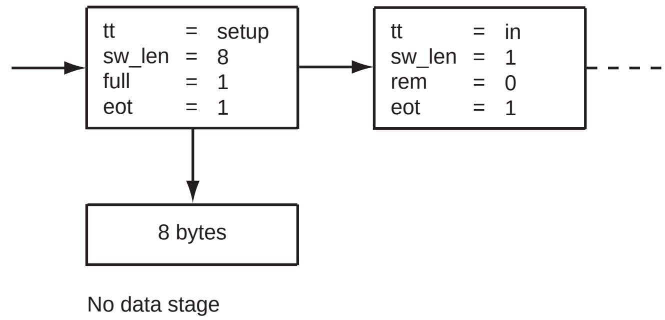


Figure 8-14 Example of a No-data Control Transfer

EP and SB Descriptors for Interrupt (INTR) Traffic

The EP descriptors for INTR traffic represent IN or OUT interrupt pipes. The technique is to run one transfer per endpoint and then change. If the endpoint has its **eof** flag set, then this signifies a *dummy* endpoint. Dummy endpoints are used to control the frequency of interrupt transfers: they must have only one SB descriptor attached, and that must be a ZOUT transfer.

ZOUT transfers are special. The ZOUT SB descriptor for periodic traffic (INTR and ISO) is not consumed and the DMA is instructed to move to the next endpoint. If the eof flag of the EP descriptor is not set, then an empty packet is sent. If the EP descriptor has its eof flag set, then the ZOUT transfer does nothing except force the hardware to stop processing interrupt traffic for that frame.

OUT transfers of INTR traffic resemble those of BULK traffic. One transfer, or an attempt to transfer, is performed in every frame (or desired period). OUT transfers of INTR traffic with different interrupt periods are complex to support.

In the first example below, the host will deliver 40 bytes in the first frame, after which it will send an empty packet in each frame. Note that if a device replies with NAK, or makes no reply, the host will attempt to re-send the packet in the next period (the next frame in this example).

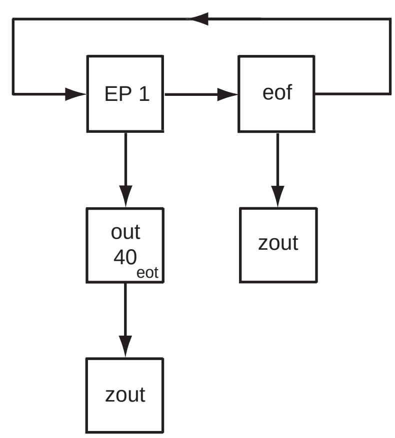


Figure 8-15 Example of Endpoint 1 (OUT) in Each Frame of INTR Traffic

In the next example below, the host will deliver 30 bytes in the first frame, then wait five frames and deliver 20 bytes. After this an empty packet will be transmitted every fifth frame. Note that if a device replies with NAK, or makes no reply, the host will attempt to re-send the packet in the next period (after five frames in this example).

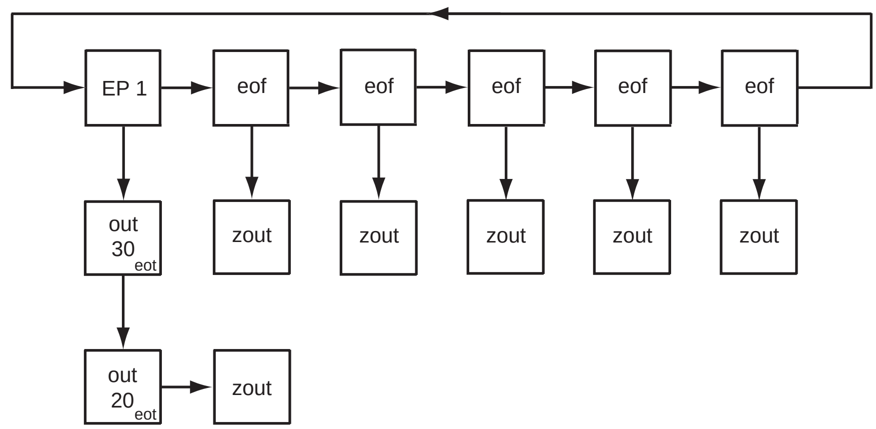


Figure 8-16   Example of Endpoint 1 (OUT) Every Fifth Frame of INTR Traffic

IN transfers of periodic traffic (INTR and ISO) are different from those of bulk and CTRL traffic. When the IN transfer is finished the SB descriptor is not consumed, which makes the IN transfer remain in the list. The first example below shows IN endpoint EP1 in each frame and IN endpoint EP2 in every second frame.

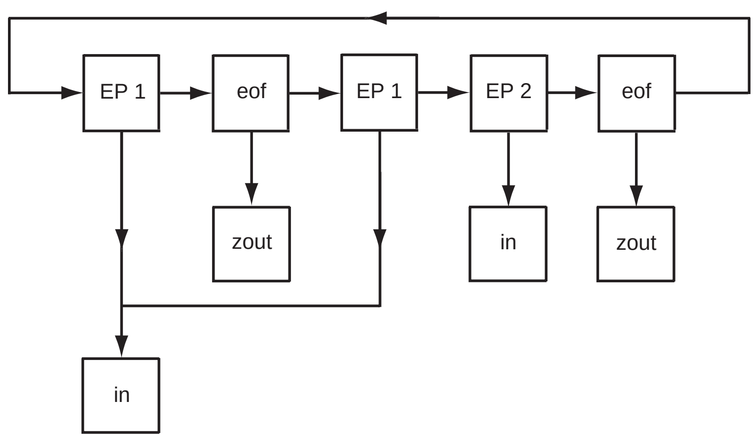


Figure 8-17   Example of Endpoint 1 (IN) Every Frame and Endpoint 2 (IN) Every 2nd Frame of INTR Traffic

The final example shows IN endpoint EP1 in every fifth frame.

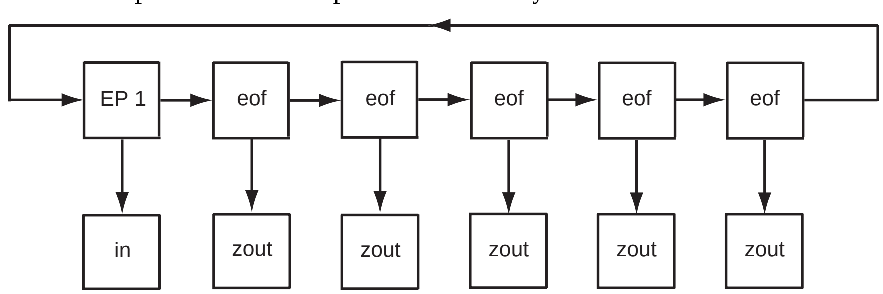


Figure 8-18   Example of Endpoint 1 (IN) Every Fifth Frame of INTR Traffic

EP and SB Descriptors for Isochronous (ISO) Traffic

Isochronous transfers are uni-directional, and the EP descriptors represent isochronous pipes. An EP descriptor with the **eof** flag set is a dummy endpoint used to mark the end of isochronous traffic. This **eof** endpoint must contain one SB descriptor with a ZOUT transfer.

For OUT transfers, only one transaction per endpoint is transmitted in each frame. If the SB list ends, then the endpoint is disabled. No re-transmissions are used. In the event of an underrun error, the USB controller forces the DMA to skip the transaction in which the underrun occurs. To make this behavior useful, it is recommended that there should be only one transaction per transfer (e.g. the data size should not be longer than the max packet size).

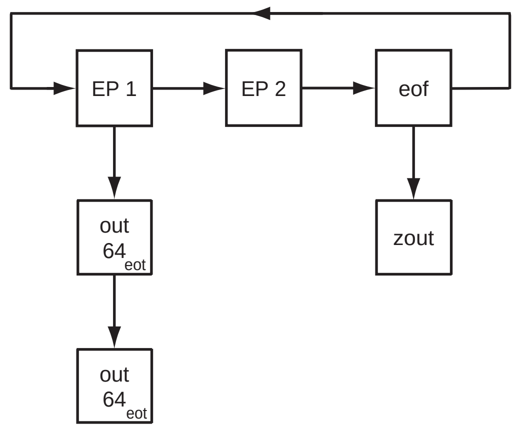


Figure 8-19 Example of ISO OUT traffic

For IN transfers, one request per endpoint is transmitted in each frame. As with interrupt traffic, the IN transfers never end. The **iso_eot** interrupt is given for all ISO IN requests.

8.5.3 Endpoint Table in Host Mode

The USB interface can handle 31 active endpoints in devices on the bus. The interface therefore includes a lookup (EP) table for translating endpoint identifiers into device/endpoint pairs. The EP table also contains status information for each valid **ep_id**. Most of this information is used by the hardware only, but some fields are used to exchange information between the hardware and the software.

The EP table contains 32 endpoint entries, each pointing at an endpoint on a device on the USB. The EP table for isochronous traffic differs slightly from that of all other traffic types: it requires fewer fields.

The EP table is accessed via three mode registers:

R_USB_EPT_INDEX is used to point to the desired entry in the EP table.

R_USB_EPT_DATA is used to read and write to the indexed entry in the EP table for all traffic types except isochronous.

R_USB_EPT_DATA_ISO is used to read and write to the indexed entry in the EP table for isochronous traffic.

After reset, all entries in the EP table are invalidated, which means that the **valid** field (bit 31), of each entry in register R_USB_EPT_DATA is cleared.

If no periodic transfer is ever scheduled, then all 32 entries in the EP table can be used. However, if interrupt or isochronous traffic are to be used, then one entry must be invalid, leaving 31 entries available.

Note 7: In software, the best way to create an invalid entry is to allocate one entry in the table; do not adjust it thereafter.

The software must never modify an entry in the EP table as long as the corresponding EP descriptor is enabled. If an entry is modified while the EP descriptor is enabled, the behavior is undefined and unpredictable. To avoid such errors a synchronization method must be used. Please refer to section 8.7 *Physical Interface* for details.

For a full definition of the fields in the EP table for all traffic types except isochronous, please refer to chapter 18.16.7 *R_USB_IRQ_MASK_CLR*.

For a full definition of the fields in the EP table for isochronous traffic, please refer to chapter 18.16.18 *R_USB_RH_STATUS*.

8.5.4 Host Mode Interrupts

A number of interrupts are generated by the USB interface: most of them indicate events in specific devices on the USB. All of the USB interrupts have the same internally-generated vector number (0x3F).

As noted in table 8-1 above, the USB interrupts are handled in four registers:

R_USB_IRQ_MASK_SET;
R_USB_IRQ_MASK_READ;
R_USB_IRQ_MASK_CLR;
R_USB_IRQ_READ.

It is also necessary for the USB interface to identify any endpoint that triggers an interrupt. Register R_EPID_ATTN is used for this purpose. Each bit in the register corresponds to an entry in the endpoint lookup table.

In summary the USB interrupts are:

iso_eof
This interrupt is triggered in response to an isochronous traffic end-of-frame flag. In Host mode the **iso_eof** interrupt occurs when the DMA reports a set **eof** flag in an isochronous EP descriptor to the USB interface. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

intr_eof
This interrupt is triggered in response to an interrupt traffic end-of-frame flag. In Host mode the **intr_eof** interrupt occurs when the DMA reports a set **eof** flag in an interrupt EP descriptor to the USB interface. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

iso_eot
In Host mode, this interrupt is triggered when an isochronous transaction is completed. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

intr_eot
In Host mode, this interrupt is triggered in response to an end-of-transfer flag when an interrupt transfer is completed. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

ctl_eot
In Host mode, this interrupt is triggered in response to an end-of-transfer flag when a control transfer is completed. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

bulk_eot
In Host mode, this interrupt is triggered in response to an end-of-transfer flag when a bulk transfer is completed. The interrupt is cleared when the **value** field in register R_USB_EPID_ATTN is read.

epid_attn
In Host mode, this interrupt is triggered whenever a significant event occurs at an endpoint. The interrupt condition is cleared when the R_USB_EPID_ATTN register is read. The events that trigger an **epid_attn** interrupt are:

invalid ep_id - occurs when the USB interface receives a transfer with an invalid endpoint identifier in the EP descriptor. It is probably caused by a programming error and is a good reason to stop the affected endpoint at least. The EP descriptor will be disabled by the hardware.

stall - not strictly an error condition but the EP descriptor is disabled nevertheless. For CTRL transfers there are special precautions to be taken by the software. The hardware does not perform any special routine for CTRL stalls. They are handled in the same way as BULK or INTR stalls.

3rd error - occurs in response to three successive transaction error in a transfer, which disables the EP descriptor. Another severe error in a transaction may also trigger this event. Inspect the EP table.

buffer ourun - a buffer overrun or underrun occurs if the memory or DMA cannot handle the traffic load. The EP descriptor is disabled.

past eof1- this event is triggered if an INTR or ISO transaction proceeds beyond the EOF1 mark in the frame.

near eof - this relatively rare event is triggered if an attempt is made to start an INTR or ISO transaction that would not fit inside the frame. It is handled in the same way as the **past eof1** event.

zout transfer - this error event is triggered if a ZOUT transfer is ever presented to any BULK endpoint. The EP descriptor is disabled.

setup transfer - this error event is triggered if a SETUP transfer is ever presented to any INTR, ISO or BULK endpoint. The EP descriptor is disabled.

sof
In Host mode, this interrupt is triggered whenever a start-of-frame flag leaves the USB interface. The interrupt is cleared when the **value** field in register R_USB_FM_NUMBER is read.

port_status
In Host mode, this interrupt signals any change in the status registers of the configured USB ports. The interrupt is cleared when these registers (R_USB_RH_PORT_STATUS_1 and R_USB_RH_PORT_STATUS_2) are read.

ctl_status
In Host mode, this interrupt indicates a change in the status of the USB interface controller. The interrupt is cleared when register R_USB_STATUS is read.

8.6 Device Mode

In Device mode, only one port can be configured for use, either port p1 or port p2. Selection of the port to be used is made by asserting fields **usb1** (bit 29) or **usb2** (bit 30) in general configuration register R_GEN_CONFIG. The act of port selection enables the USB interface, which is entirely inoperative prior to this.

All operations at the enabled port are handled by the root hub under the control of the **port_sel** (bits 7 and 6) and **port_cmd** (bits 5 and 4) fields in register R_USB_COMMAND_DEV. The status of the enabled port is read in register R_USB_RH_PORT_STATUS_1 if port p1 is in use, or R_USB_RH_PORT_STATUS_2 for port p2.

Root hub status register R_USB_RH_STATUS is not used in Device mode.

8.6.1 USB Controller Commands in Device Mode

Register R_USB_COMMAND_DEV contains fields for commanding the root hub and the USB controller. All these fields must be written in one operation.

Each time a write operation is performed to register R_USB_COMMAND_DEV, a command interpretation is triggered. This sets the **busy** field (bit 7) in register R_USB_STATUS. When the USB controller has executed the command, the **busy** field is cleared. The current state of the USB interface is read in main status register R_USB_STATUS.

The only commands that can be issued to the USB controller in Device mode are:

nop - (no operation)
The enabled port can be commanded without issuing any commands to the USB controller. The **nop** command is issued by setting field **ctrl_cmd** (bits 2 to 0) in register R_USB_COMMAND_DEV to the value 0x0.

deconfig
This is an emergency stop command that immediately deconfigures the entire USB interface, returning it to the condition that immediately succeeds a reset. The state of the USB controller changes to UNCONFIGURED. The **deconfig** command is issued by setting the **ctrl_cmd** field in R_USB_COMMAND_DEV to the value 0x2.

host_config
If the USB controller is in the UNCONFIGURED state, this command configures it as a host controller. The **host_config** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND_DEV to the value 0x3.

dev_config
If the USB controller is in the UNCONFIGURED state, this command configures it as a device controller. The **dev_config** command is issued by setting the **ctrl_cmd** field in register R_USB_COMMAND_DEV to the value 0x4.

8.6.2 USB Port (Root Hub) Commands in Device Mode

In Device mode, commands to the root hub are given in the **port_cmd** field of register R_USB_COMMAND_DEV. They are issued to the port selected in the **port_sel** field (bits 7 to 6) of the command register as follows:

active
Enable the selected port. This must be done after the **dev_config** command has been issued to the USB controller. The **active** command is issued by setting the **port_cmd** field in register R_USB_COMMAND_DEV to the value 0x0.

passive
Disable the selected port. Traffic on the bus is ignored and none is sent from the device. The **passive** command is issued by setting the **port_cmd** field in register R_USB_COMMAND_DEV to the value 0x1.

wakeup
Drive a K-state of 3 ms in length on the bus to request remote wakeup. This command is valid only if the device is in a suspended mode. The **wakeup** command is issued by setting the **port_cmd** field in R_USB_COMMAND_DEV to the value 0x3.

8.6.3 USB Data Structures in Device Mode

In Device mode, the USB controller can handle four IN endpoints and 12 OUT endpoints. Note that the terms IN and OUT refer to the direction at the USB host; in other words, an IN Transfer is outgoing from the device, and an OUT transfer is incoming to the device. DMA channel 8 is used for outgoing traffic from the USB device to the host and is divided into four sub-channels, one for each IN endpoint.

The data structures for the sub-channels of DMA channel 8 are organized as four linked lists of linked lists, where the first list (the EP list), always points to itself. DMA sub-channel 8.0 is used as a control endpoint, while sub-channels 8.1to 8.3 can be used for any traffic type.

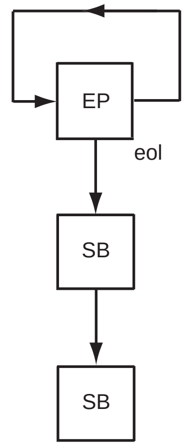


Figure 8-20 DMA List Structure for USB Device (IN Endpoint)

ETRAX 100LX supports 12 endpoints for incoming traffic from the USB host (OUT traffic). DMA channel 9 handles this traffic, all of which is placed in the FIFO buffer of DMA channel 9. This channel behaves in the same way as any other DMA channel in ETRAX 100LX, but with additional status flags.

DMA Descriptors for OUT Transfers in Device Mode

The format of a DMA descriptor for an OUT transfer (incoming traffic to the device) is shown below. This format is used by DMA channel 9, and it is recommended that all DMA descriptors for USB are 32-bit aligned.

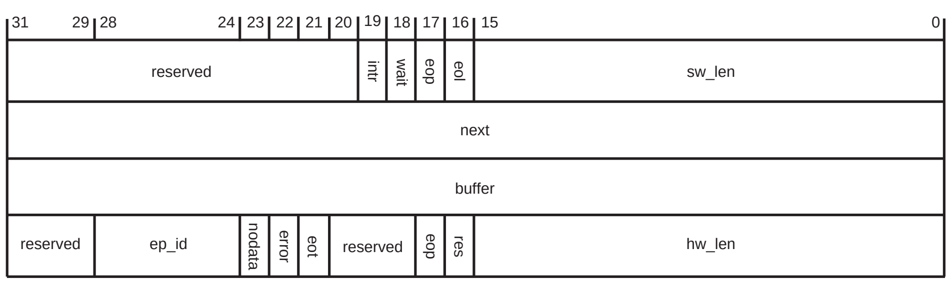


Figure 8-21 Format of DMA Descriptor

For a detailed description of the DMA descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB.*

EP Descriptors in Device Mode

Endpoint descriptors are mainly used to store the **ep_id** number but, in USB Device mode, the **ep_id** number is fixed as follows:

Sub-Channel	ep_id
8.0	0
8.1	1
8.2	2
8.3	3

The format of an EP descriptor, used by DMA channel 8, is shown below. There is only one EP descriptor per sub-channel. All EP descriptors must be 32-bit aligned.

31–29	28–24	23–21	20	19	18	17	16	15–0
reserved	ep_id	reserved	enable	intr	res	eof	eol	hw_len
sub								
nep								

Figure 8-22 Format of an EP Descriptor

Note that the eof bit is not used in Device mode, and should be set to 0. For a detailed description of the EP descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB.*

SB Descriptors in Device Mode

In Device mode, the SB descriptors for the sub-channels in DMA channel 8 contain status and pointer fields to the data that will be transmitted from the device to the USB host. The format of an SP descriptor in a sub-channel is illustrated below.

31–30	29–24	23	22	21–20	19	18	17	16	15–0
res	rem (not used)	res	full	tt	intr	res	eot	eol	sw_len
next									
buffer									

Figure 8-23 Format of SB Descriptor in Device Mode

Note that the rem field is not used in Device mode, and should be set to 0. In Device mode, the only transfer types allowed in the tt field are ZOUT and OUT. For a detailed description of the SB descriptor refer to chapter 7.4.3 *DMA Descriptor Format for USB.*

8.6.4 EP Table in Device Mode

The EP table is used in Device mode as well as Host mode, and is accessed through mode registers:

- R_USB_EPT_INDEX which is used to point to the desired entry in the EP table.
- R_USB_EPT_DATA_DEV which is used to read and write to the indexed entry in the EP table.

The EP table also contains status information for each valid **ep_id**. 8-2 below shows how the rows in the EP table are used:

To fill rows 0 - 11 and 16 - 19, the register macros for R_USB_EPT_DATA_DEV could be used.

The data toggle bit for an endpoint is only initialized during a setup transaction.

ep_id	Description
0 - 11	Configuration for USB out traffic - incoming traffic to the device. All data received will end up in DMA channel 9, and there are 12 endpoints available. ep_ids 20 - 31 are used to know the transfer size of USB out traffic: ep_id 20 is used by ep_id 0 ep_id 21 is used by ep_id 1 ep_id 22 is used by ep_id 2 ep_id 23 is used by ep_id 3 ep_id 24 is used by ep_id 4 ep_id 25 is used by ep_id 5 ep_id 26 is used by ep_id 6 ep_id 27 is used by ep_id 7 ep_id 28 is used by ep_id 8 ep_id 29 is used by ep_id 9 ep_id 30is used by ep_id 10 ep_id 31is used by ep_id 11
12 - 15	Reserved
16 - 19	Configuration for USB in traffic - outgoing traffic from the device. There are four endpoints are available, one for each DMA channel 8 subchannel.
20 - 31	Counters for ep_ids 0 - 11. Bit [31:16] is the transfer size Bit [15:0] is the number of bytes remaining in the ongoing transfer. Note that for a control endpoint, the counter is loaded by hardware with the transfer size received within the setup packet.

Table 8-2 EP table usage

For a full definition of the fields of the EP table in Device mode, please refer to chapter 18.16.19 *R_USB_RH_PORT_STATUS_1*.

8.6.5 Device Mode Interrupts

The USB interrupts in Device mode are:

out_eot
This interrupt is triggered by an end-of-transfer flag on any of the twelve OUT endpoints.

ep3_in_eot, ep2_in_eot, ep1_in_eot
These interrupts are triggered by an end-of-transfer flag on IN endpoints numbered 1 to 3 respectively.

ep0_in_eot
This interrupt is triggered by an end-of-transfer flag on IN endpoint number 0 (control).

epid_attn
In Device mode, this interrupt is triggered in response to an endpoint overrun or underrun condition. The interrupting endpoint is identified in register R_USB_EPID_ATTN and the error condition can be read in register R_USB_STATUS.

sof
In Device mode, this interrupt is triggered each time the frame timer reaches 0. The frame timer interval is set in R_USB_FM_INTERVAL, and can be synchronized with the frame interval of the host by using the information in R_USB_FM_NUMBER_DEV.

port_status
In Device mode, this interrupt signals any change in the status of the USB port.

ctl_status
In Device mode, this interrupt indicates a change in configuration.

8.7 Physical Interface

The physical interface of the USB is divided into two main parts: data transmission and power management. These differ slightly in Host mode and Device mode.

8.7.1 Data Transmission

The data transmission interface is compatible with Philips USB transceiver type PDIUSBP11A. It is almost identical for Host and Device mode, with minor differences in the protocol handling. There is also a difference on the outside of the transceiver, concerning the pull-up and pull-down for device speed indication.

In host mode, D+ and D- are individually pulled down with 15K Ohm.

In Device mode, a 1.5K Ohm resistor is used for device speed selection. If a full-speed device is built, the resistor should be a pull-up on the D+ line. For a low-speed device, the resistor should be a pull-up on the D- line.

8.7.2 Power Management

USB power management is not implemented in the ETRAX 100LX USB interface. Since power management is slow and rare, it can be implemented in software using general port PA for overcurrent sensing, and any generic I/O pin for power on/off control. The fact that general port PA can be configured to generate interrupts is a useful feature in this regard.

8.7.3 Hardware Reset

Whenever the USB interface is not configured in register R_GEN_CONFIG, then the USB interface reset is active and the interface is entirely frozen. Writing to the USB registers has no effect, and reading from the registers may produce unpredictable results. Immediately after reset the USB interface is in a passive state. The USB ports are off, in the sense that their output enable signals are inactive.

Note 8: One way to perform a hardware reset of the USB interface is to deconfigure the interface in register R_GEN_CONFIG and then reconfigure the interface.

It is the responsibility of the software to manage USB power during startup of the USB interface. This is done outside the USB interface as noted in 8.7.2. above.

8.8 Procedures

8.8.1 Configuring the USB Interface for Host Mode

Setting the General Configuration

Set the **usb1** (bit 29) and/or **usb2** (bit 30) fields in register R_GEN_CONFIG to configure either or both of the two available USB ports. This initializes the USB interface and causes a hardware reset of the USB interface to be performed.

Configuring the Interface

Set the **ctrl_cmd** field (bits 2 to 0) in register R_USB_COMMAND to the value host_config (0x3). This issues the **host_config** command to the USB interface.

8.8.2 Starting and Stopping the Host Mode

When configured, the USB interface eventually reaches the HOST_MODE state, where it remains until at least one port is enabled. When a port is enabled, the HOST_STARTED state is attained and traffic processing can be started by issuing the **host_run** command to the USB controller. This sets the USB interface to the HOST_RUNNING state.

Starting USB Traffic Processing

Set the **ctrl_cmd** field in register R_USB_COMMAND to the value **host_run (**0x6). This issues the **host_run** command to the USB controller.

Stopping USB Traffic Processing

Set the **ctrl_cmd** field in register R_USB_COMMAND to the value **host_stop** (0x7). This issues the **host_stop** command to the USB controller.

All traffic processing is suspended but frame generation continues. The traffic for the current frame is finished before the state changes to HOST_STARTED. The traffic can be started again without any special actions as long as the data structures are intact. However it should be noted that the devices may have become unstable during the stoppage time.

8.8.3 Starting and Stopping Traffic in Host Mode

This is related to the start/stop of traffic processing. In principle, traffic can be forcibly stopped by commanding the USB controller from the HOST_RUNNING state to the HOST_STARTED state. However a clean shutdown may leave devices in a safer condition than a forced shutdown.

Stopping Each Traffic Type with a Clean Procedure

1. Withdraw any pending transfers. This may require the temporary disabling of the endpoint in order to inspect and manipulate the SB descriptor lists.
2. Wait for the DMA sublist to stop. When the endpoints exhaust one by one, they are automatically disabled by the USB interface. When all endpoints are disabled, the DMA sub-channel will stop.

3 When all sub lists have stopped, issue the **host_stop** command to the USB controller by setting the **ctrl_cmd** field in register R_USB_COMMAND to the value **host_stop** (0x7).

Restarting Traffic After a Stop

1 Set the **ctrl_cmd** field in register R_USB_COMMAND to the value **host_run** **(**0x6). This issues the **host_run** command to the USB controller.

2 Set up the new transfers to be processed. All traffic was withdrawn before the traffic was stopped and therefore all SB lists are empty.

8.8.4 Managing EP Descriptor Lists in Host Mode

All EP descriptor lists must have a disabled dummy EP descriptor with the eol flag set. To create a new EP descriptor list it is necessary to create the first descriptor, link the first descriptor to the dummy EP descriptor, and set the **eol** flag. This must be done before the DMA sub-channel is started. The list will resemble the following diagram:

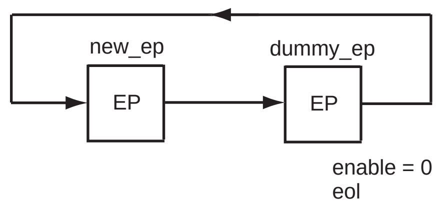


Figure 8-24 New EP Descriptor List in Host Mode

Inserting an EP Descriptor into an Existing List

1 Create a new EP descriptor, here named **new_ep**. It is recommended that all values in the new descriptor should be set to zero.Locate the point in the list where the **new_ep** is to be inserted. The descriptor that precedes the insertion point is here named **before_ep** and the descriptor that succeeds the insertion point is named **after_ep**.

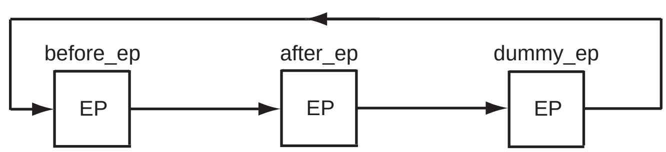


Figure 8-25 Existing Descriptor List

2 Point the new endpoint at the **after_ep** descriptor. This must be done before the next step.

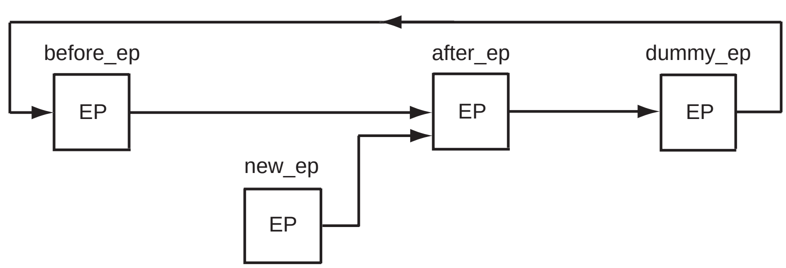


Figure 8-26 Pointing the New Endpoint at the Succeeding Descriptor

3 Point the **before_ep** at the **new_ep** descriptor. To ensure that the hardware does not detect a partially-updated new endpoint, this must be done in one single 32-bit write operation.

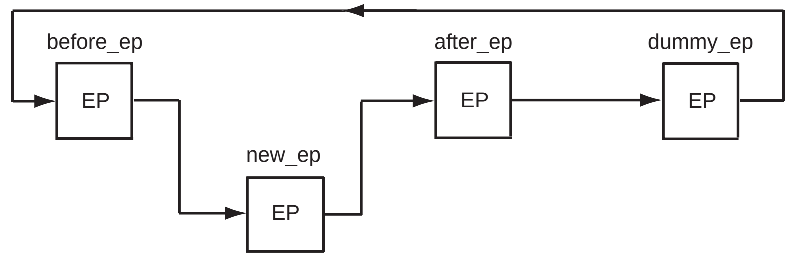


Figure 8-27 Pointing the Preceding Descriptor at the New Endpoint

Removing an EP Descriptor from the List

1 Bypass the descriptor by changing before_ep.next to old_ep.next. This must be done in one single 32-bit write operation.

2 Check that the value in R_DMA_CH8_SUBx_EP is not equal to old_ep.

3 If the values are not the same it is safe to remove the old_ep.

4 If these two values are equal, set the interrupt bit intr in the *dummy* descriptor and wait for the dma8_subx_descr interrupt.

5 At the dma8_subx_descr interrupt, remove old_ep, and clear the interrupt bit in the dummy descriptor.

6 Acknowledge the interrupt by writing to R_DMA_CH8_SUBx_CLR_INTR.

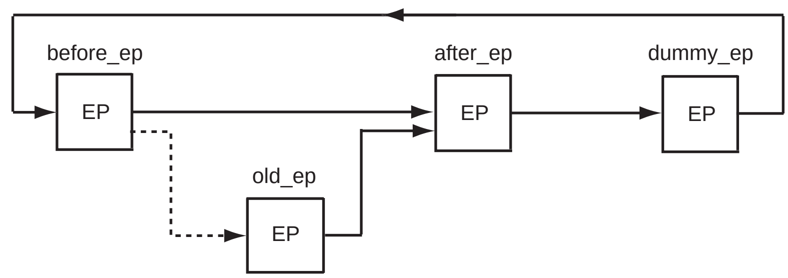


Figure 8-28 Removing the EP Descriptor

8.8.5 Managing SB Descriptor Lists in Host Mode

It is possible to insert descriptors in the middle of an SB descriptor list, but advanced disable/enable EP descriptor manipulation with list traversal is necessary to find the intersections between transfers. New traffic must not be inserted in the middle of a transfer because it would confuse the USB hardware and produce unpredictable results.

The preferred method of inserting descriptors is to append new traffic at the end of the list, as described below. It is recommended an entire transfer should be inserted in one operation because this simplifies other aspects of the operation.

1 Find the last SB descriptor in the list.

2 Append the new descriptor to the list.

3 Clear the **eol** flag in the last descriptor.

4 Check whether the EP descriptor at the head of the list is enabled or disabled. If it is enabled, then the procedure is concluded. Otherwise go to step 5.

5 Check whether an error disabled the EP descriptor, taking care not to lose any information from the interrupt system. If there was an error condition, proceed with step 6. Otherwise go to step 7.

6 Correct the error condition, which may necessitate a permanent stop of the endpoint. This must be handled in software, which must correctly withdraw all transfers in the SB descriptor list, including the newly-inserted transfer. When this has been done, restart the endpoint if possible.

7 Check if the sub field in the EP descriptor is pointing to the new SB descriptor. If so, the procedure is completed; otherwise, go to step 8.

8 Update the EP descriptor. There was no error and the EP descriptor was disabled, therefore the SB descriptor list was exhausted. The sub field in the EP descriptor must be pointed at the new SB descriptor.

9 Enable the EP descriptor to conclude the procedure.

When removing descriptors from the SB descriptor list it is very important not to confuse DMA, which could lead to unpredictable events in the USB interface. For more information, please refer to the above section 8.8.4 *Managing EP Descriptor Lists in Host Mode*.

Removing SB Descriptors from the List

1 Disable the endpoint, see 8.8.4 *Managing EP Descriptor Lists in Host Mode*.

2 Find the start and stop of the transfer to retire. If the endpoint was disabled due to an error, the error must also be corrected.

3 Remove the transfer to retire.

4 If the list is not empty, re-enable the endpoint.

8.8.6 Managing the EP Table in Host Mode

Connecting an EP Descriptor to an EP Table Entry

1 Allocate an EP table entry. The entry number is the **ep_id.**

2 Write the **ep_id** into the **value** field of register R_USB_EPT_INDEX and set the **ep_id** field in the EP descriptor.

3 Clear the EP table entry in register R_USB_EPT_DATA.

4 Set the device and endpoint parameters in register R_USB_EPT_DATA.

5 Set the **valid** field (bit 31) in the EP table entry.

6 Insert the EP descriptor into an EP list.

7 Enable the EP descriptor.

Disabling an Endpoint and Accessing the EP Table Entry

Follow the procedure set forth in section 8.8.4 *Managing EP Descriptor Lists in Host Mode*. When the EP is safely disabled, the entry in the EP table could also be modified.

Re-enabling the Endpoint

1 Set the enable bit in the EP descriptor.

2 Check that the DMA is still running. If it has stopped for some reason (e.g. no more data to process), then write a start command to the appropriate DMA command register.

8.8.7 Managing the DMA Channel 9 Descriptor List

This is largely the same as managing any DMA list for incoming traffic. A typical USB DMA descriptor list structure resembles Figure 8-29 below.

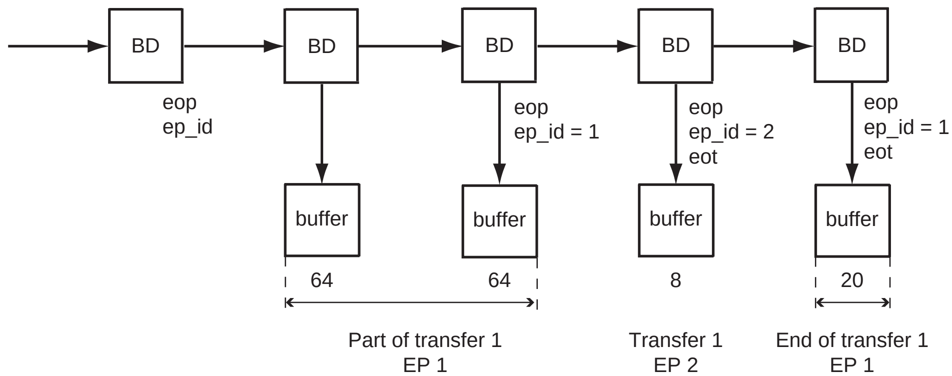


Figure 8-29 Typical USB DMA Descriptor List

After reception of a transfer (one packet), the **ep_id** and the **eop** field are written to the DMA descriptor in DMA channel 9. If the transfer is an empty packet, a dummy byte is written and the **nodata** bit is set.

Observe that a packet ends in the buffer that is flagged eop, and the next packet begins in the immediately succeeding buffer.

The software must manage the demultiplexing of packets to the correct pipe, and the cleanup after an unrecoverable transaction error.

8.8.8 Managing the Root Hub

As previously noted, the root hub manages the low level details of the USB connect/disconnect, speed detection, reset and suspend/resume signalling. The root hub also detects faulty signalling and babbling devices.

Section 8.8.3 *Starting and Stopping Traffic in Host Mode*, describes the command sequences to the USB controller. The command sequences to reset and enable a USB port are discussed here.

The root hub has dedicated registers to disable the USB ports. These registers are R_USB_PORT1_DISABLE and R_USB_PORT2_DISABLE. They must be configured before a USB port can be used.

Writing to the Disable Registers

Set the **disable** field (bit 0) in register R_USB_PORT1_DISABLE to the value no (0x1). This ensures that USB port p1 can be enabled. Set the **disable** field (bit 0) in register R_USB_PORT2_DISABLE to the value no (0x1). This ensures that USB port p2 can be enabled.

Detecting a Port Event

1 Wait for the USB interrupt. If register R_USB_IRQ_READ indicates a **port_status** interrupt, go to step 2. Otherwise wait until an interrupt occurs.

2 Handle the event. Ensure that the software can remember the values in these registers in order to detect the condition that has changed.

Some events also trigger a ctl_status interrupt. For instance a command changing a port from **reset** to **enabled** condition may set the USB controller from the HOST_MODE state to the HOST_STARTED state. Also note that the last port to change from **enabled** to **disconnected** or **reset** condition will return the controller to the HOST_MODE state.

If a port event indicates that a port became connected, it is appropriate to reset the port as follows:

1 Issue a reset command to the port by writing to the R_USB_COMMAND register.

2 Wait for the reset to complete. Wait for the port enabled event.

3 Notify the upper software that an enabled event occurred at the port.

The sequence above shows a command to one port (i.e. the root hub). This is the best way to command the root hub. It is possible to issue commands to the USB controller and the root hub, but there is a potential for error.

Many operations on the root hub follow the same pattern:

1 Perform an event.

2 Wait for its completion.

3 Report the change in status.

Disabling a Port

1 Set the **disable** fields (bit 0) in register R_USB_PORT1_DISABLE or R_USB_PORT2_DISABLE to the value yes (0x0). This forces the respective port to recognize a disconnect event and change the port status accordingly.

2 Wait for the port status to change to disconnected.

8.8.9 Managing USB IN Traffic in Device Mode

As previously noted, in Device mode the USB controller has 4 IN endpoints. The traffic for those endpoints are scheduled in the sub-channels of DMA channel 8. Because of the endpoint numbers (0, 1, 2 and 3), it is necessary to use endpoint 0 as the control channel. Any of the other endpoint (1, 2 and 3) can carry bulk, interrupt or isochronous traffic.

Creating an IN Endpoint

1 Select an endpoint (0, 1, 2 or 3). In this example, endpoint 1 is used.

2 Create an EP list for the corresponding DMA sub-channel. The list comprises one EP descriptor pointing at itself.

3 Create SB descriptors describing the data that will be sent. In this example 321 bytes will be sent, with a buffer size of 256 bytes.

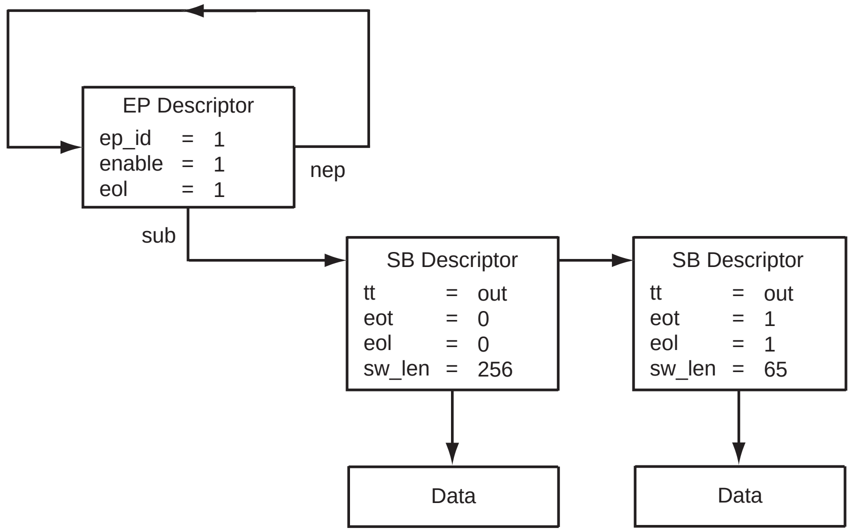


Figure 8-30 Creating an IN Endpoint

4 Fill information into the EP table. Write to the corresponding row in the EP table (row 17 since endpoint 1 has been chosen). This enables the endpoint and sets its parameters.

5 Start the DMA sub-channel by setting the **cmd** field (bit 0) in register R_DMA_CH8_SUB1_CMD to the value **start (**0x1).

If the USB host submits a request to this endpoint, the device will respond with data. When all data are sent, the sub-channel becomes disabled.

If the USB device controller receives more requests to this endpoint, it responds with NAK until the software places more data into the sub-channel and restarts it.

Appending Traffic to an IN Endpoint

1 Create an SB descriptor describing the data packet to be sent. In this example a USB transfer of 100 bytes in length will be appended. Set the **eot** and **eol** flags in this descriptor.

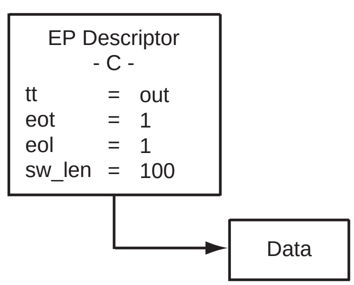


Figure 8-31 Creating an SB Descriptor for Appending Traffic to an IN Endpoint

2 Move the next pointer in the last SB descriptor (B) to point at (C).

3 Clear the **eol** flag in descriptor B. It is important that the next pointer is updated before the **eol** flag is cleared.

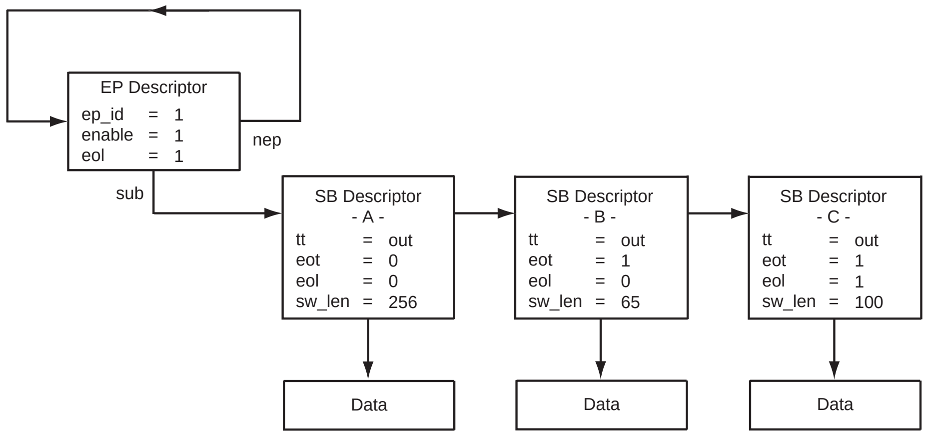


Figure 8-32 Clearing the eol Flag in Descriptor B

4 Check whether the sub-channel is stopped. The hardware stops a sub-channel only when the **eol** flag is reached. If the sub-channel is running, then the procedure is concluded. Otherwise go to step 5.

5 Check the sub-pointer in the EP descriptor. If this pointer is not directed at the new SB descriptor (C), then point it there and start the sub-channel. If the pointer in the EP is already pointing at the new SB, then that traffic has been sent and the sub-channel need not be started.

Stalling Traffic on an IN Endpoint

Set the stall bit in the corresponding row in the EP table. This forces the USB hardware to respond with a stall on all requests from the host to that endpoint. The software must disable stall mode to make this endpoint operate again. This should occur after a configuration event on the control channel (channel 0). It is not recommended that endpoint 0 should be stalled.

Removing Traffic from an Endpoint

1 Stop the DMA sub-channel by setting the **cmd** field (bit 0) in the appropriate register R_DMA_CH8_SUBn_CMD to the value **stop** (0x0).

The hardware will transmit the ongoing transaction. If the transmission of a packet has started, the hardware will attempt to finish the transmission. However the relatively abrupt stop of the DMA may cause an underrun that forces the hardware to abort the packet. This is signalled in the same way as an ordinary underrun condition.

2 Wait for the transaction to end. This cannot be observed by software, but the transaction cannot be longer than one USB frame. It is, therefore, possible to wait for one frame to have passed. This could be done by using the sof interrupt which is given once every frame.

3 Modify the list if necessary. Then start the sub-channel.

Error Conditions When Managing USB IN Traffic

All USB errors are handled by hardware, and the only error that can be reported is underrun. The underrun error is indicated in register R_USB_STATUS, and the number of the endpoint that caused the error can be read in register R_EPID_ATTN.

If an isochronous endpoint encountered the underrun error, the hardware will abort the transaction and then disable the endpoint by clearing the valid bit at the corresponding row in the EP table. If a control, bulk or interrupt endpoint encountered the underrun error, the hardware only aborts the transaction, leading to a new request from the USB host.

8.8.10 USB OUT Traffic in Device Mode

As previously mentioned, all incoming traffic from the 12 OUT endpoints is handled in DMA channel 9. Endpoint 0 is always a control endpoint and the other endpoints can be bulk, interrupt or isochronous. All OUT endpoints require two entries in the EP table, one describing the endpoint and one containing the number of bytes in the transfer.

When constructing software for the device it is advisable to ensure that there are always free descriptors/buffers in DMA channel 9. This can be achieved by creating a list with interrupts enabled at certain points in the list - helping the software to free up descriptors. DMA channel 9 will stop if no descriptors remain, resulting in an overrun in the USB hardware.

USB Bulk and Interrupt OUT Traffic in Device Mode

As for all USB OUT transfers, the USB hardware must be informed of the expected transfer length. This is the maximum length for the transfer. The transfer could be shorter, but this must be signalled from the host by sending a short packet (not **max_packet_size**) or an empty packet.

If the transfer length is not known, the counter must be loaded with a large number (i.e 0xffff) and then reloaded with this value before it reaches zero. For bulk and interrupt traffic, the **max_packet_size** is 64 =>. The counter must be reloaded after ~1000 packets and, at the most, there can be 19 of these packets per frame (per 1 ms).

Another approach to an unknown transfer length is to load the counter with a multiple of the **max_packet_size** (i.e 0xffc0). The hardware will thus report *end-of-transfer* after 1023 packets but, if the actual end-of-transfer has not occurred, the host will continue to send data and the device hardware will receive the data.

After reception of a transaction (one packet), the **ep_id** and the **eop** field are written to the DMA descriptor in DMA channel 9. If the transaction is an empty packet, a dummy byte is written and the **nodata** bit is set.

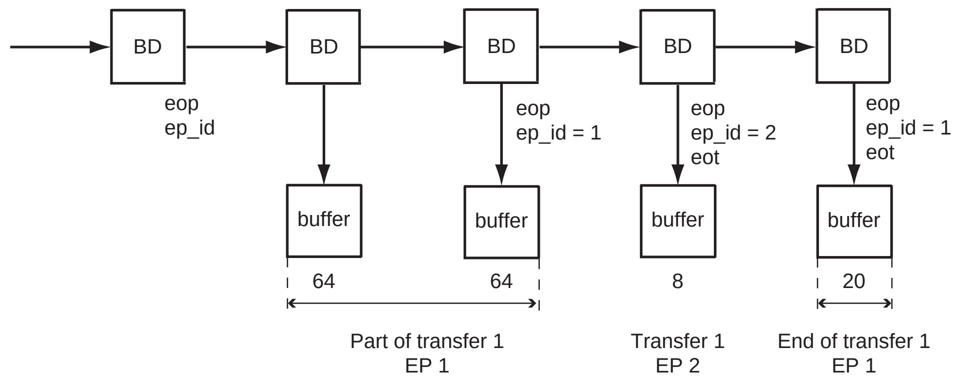


Figure 8-33 Typical USB DMA descriptor List

As mentioned before, the end-of-transfer can be signified by the receipt of:

- Exactly the number of previously agreed bytes
- A short packet
- An empty packet

At the end-of transfer, an interrupt is signalled (note that this interrupt is the same for all 12 OUT endpoints), and the **eot** bit is written in the DMA descriptor that contains the last data for that transfer.

USB Isochronous OUT Traffic in Device Mode

In Device mode, isochronous OUT traffic operates in almost the same way as bulk and interrupt traffic. The difference is that each transaction on an isochronous channel is a transfer. Consequently the transfer length required by hardware (from the EP table), must be set to the same value as **max_packet_size**.

Error Conditions for USB OUT Traffic in Device Mode

The only error not handled by the USB hardware is overrun. The overrun error is reported in register R_USB_STATUS, and the number of the endpoint that caused the error can be read in register R_EPID_ATTN. If an endpoints overruns it does not acknowledge that transaction.

8.8.11 USB Control Traffic in Device Mode

Control traffic uses one OUT endpoint and one IN endpoint. With three exceptions, they act in the same way as other endpoints. These exceptions are:

- The transfer length of the OUT endpoint (traffic to device), is handled by hardware
- IN transfers are aborted if the host issues a new setup packet
- IN transfers are considered done when the status packet arrives

The transfer length of the OUT endpoint (traffic to device), is handled by hardware

For a setup packet, the hardware always loads the EP table with the value 8 and, after reception of the first DATA0 packet, the EP table is loaded with the expected transfer length. This is the two last bytes in that packet.

IN transfers are aborted if the host issues a new setup packet

If the host first issues a Control read command and then, before all data are read by the host, it issues another Control read command, then the hardware will skip the first transfer in the IN endpoint.

IN transfers are considered done when the status packet arrives

To support the requirements of section 8.5.2.3 of the revision 1.1 of the USB specification, the IN transfer is skipped if the status packet arrives before the IN transfer is complete. Consequently, there will be no **eot** interrupt on that endpoint.

9 NETWORK INTERFACE

The ETRAX 100LX includes an on-chip Fast Ethernet controller.

The ETRAX 100LX network interface supports 10 and 100 Mbps Ethernet (IEEE 802.3 and Ethernet II) protocols. With 10 Mbps, the physical interface can be configured to be compatible with either 802.3 MII or SNI (National Semiconductor DP8391 compatible interface). 100 Mbps is only supported over 802.3 MII.

The network interface has the following features:

- 10 MBit/100 MBit MII Interface
- 10 MBit Serial Network Interface (SNI)
- Support for full duplex including pause frames
- 2 simultaneous station addresses

9.1 The Ethernet II and IEEE 802.3 Standards

There are some minor differences between the IEEE 802.3 standard and Ethernet II regarding frames. Figure 9-1 below shows the frame format for the Ethernet II standard and the frame format for the IEEE 802.3 standard.

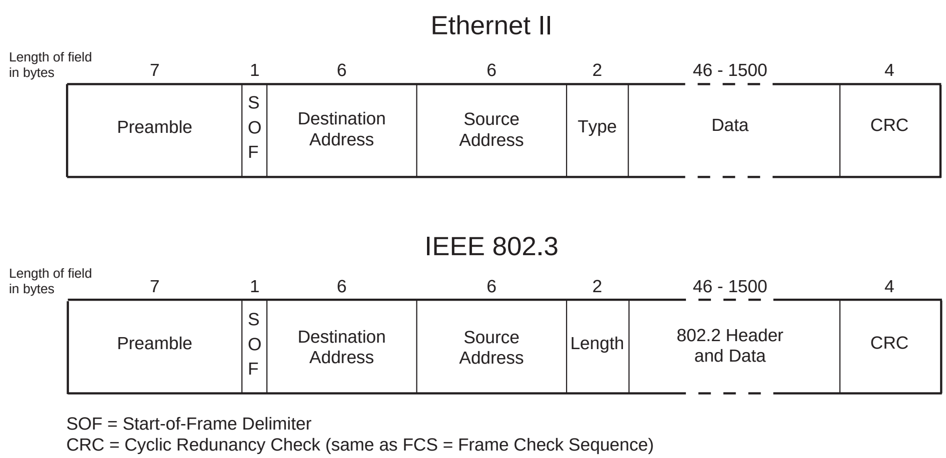


Figure 9-1 A description of the frames in Ethernet and the IEEE 802.3 standard

Table 9-1 below gives short explanations of the frames' contents:

Frame fields	Description	Length (bytes)
Preamble	Provides synchronization. Consists of 7 bytes of the bit pattern "10101010".	7
Start-of-Frame Delimiter (SOF)	Provides framing. Contains the bit pattern "10101011".	1
Destination Address	The address of the destination station. The first bit indicates: 0 = individual; 1 = group address	6
Source Address	The address of the sending station.	6
Length or type	Specifies the length or type of the data field.	2
Data Pad	The data content. If the actual data is less than 46 bytes, pad bytes consisting of zeros are added to it up to the required minimum of 46 bytes.	46 - 1500 0 - 46
CRC	Used for error checking at the destination station. (CRC = Cyclic Redundancy Check)	4

Table 9-1 Ethernet frame contents

9.2 Network interface registers

Table 9-2 below provides a brief description of the network interface registers. For more detailed information, see 18.9 *Network Interface Registers*.

Register	Function
R_NETWORK_SA_0	A 32-bit wide write only register which contains the bit address [31:0] of station address MA0.
R_NETWORK_SA_1	A 32-bit wide write only register which contains the bit address [15:0] of station address MA1, and the bit address [47:32] of station address MA0.
R_NETWORK_SA_2	A 32-bit wide write only register which contains the bit address [47:16] of station address MA1.
R_NETWORK_GA_0	A 32-bit wide write only register which contains the bit [31:0] of the group address table.
R_NETWORK_GA_1	A 32-bit wide write only register which contains the bit [63:32] of the group address table.
R_NETWORK_REC_CONFIG	A 32-bit wide write only register for configuration of the Ethernet receiver.
R_NETWORK_GEN_CONFIG	A 32-bit wide write only register for internal loop back, setting the frame format, SNI and MII mode selection, and enabling/disabling of the network controller.
R_NETWORK_TR_CTRL	A 32-bit wide write only register for controlling the Ethernet transmitter.
R_NETWORK_MGM_CTRL	A 32-bit wide write only register for controlling the management interface.
R_NETWORK_STAT	A 32-bit wide read only register for receiver pin status, transmitter error status, and management data.
R_REC_COUNTERS	A 32-bit wide read only register containing receiver error counters.
R_TR_COUNTERS	A 32-bit wide read only register containing transmitter error counters.
R_PHY_COUNTERS	A 32-bit wide read only register containing sqe_test_error and carrier_loss counters.

Table 9-2 Network interface registers

9.3 Network Interface Configuration

The network interface for the ETRAX 100LX is configured in the registers R_NETWORK_REC_CONFIG and R_NETWORK_GEN_CONFIG. In R_NETWORK_GEN_CONFIG, the following configuration possibilities are available:

- Enable/disable the network interface
- Physical interface mode, IEEE 802.3 MII or DP 8391 compatible SNI (see 9.4)
- Frame type and access protocol mode (see 9.6.2 and 9.6.3)

By setting the loopback field of R_NETWORK_GEN_CONFIG, the network interface can also be set in an internal loopback mode where the transmit data is fed directly to the receiver. The data speed in this mode is 100 Mbps.

The R_NETWORK_REC_CONFIG register contains receiver specific configurations (see 9.5). This register also contains the duplex field, which selects between half and full duplex operation.

In full duplex mode, the receiver is not turned off during transmission, and the transmitter ignores the COL and CRS signals. Full duplex flow control is also turned on. This enables the network transmitter of the ETRAX 100LX to automatically react on 802.3x PAUSE frames by temporarily stopping the transmission.

The full duplex mode can also be used for external loopback tests.

Note 1: Unless the transceiver has successfully negotiated full duplex, half duplex should be selected.

9.4 Pin Usage in MII and SNI Modes

Solder Ball	Direction	Name	MII Usage	SNI Usage
Y11	in/out	mdio	Management data.	General I/O.
W11	out	mdc	Management clock.	General output.
V11	out	txdata0	Data out, bit 0.	Data out.
U11	out	txdata1	Data out, bit 1.	General output.
Y12	out	txdata2	Data out, bit 2.	General output.
W12	out	txdata3	Data out, bit 3.	General output.
V12	out	txen	Transmit enable.	Transmit enable.
U12	out	txer	Transmit error/ 25 MHz clock/ Address recognized.	General output.
Y13	in	crs	Carrier sense.	Carrier sense.
W13	in	col	Collision.	Collision.
V13	in	txclk	Transmit clock.	Transmit clock.
Y14	in	rxer	Receive error.	General input.
W14	in	rxclk	Receive clock.	Receive clock.
Y15	in	rxdv	Data in valid.	Not used.
V14	in	rxdata0	Data in, bit 0.	Data in.
W15	in	rxdata1	Data in, bit 1.	General input.
Y16	in	rxdata2	Data in, bit 2.	General input.
U14	in	rxdata3	Data in, bit 3.	General input.

9.5 Receiver Logic Functions

The receiver logic of the ETRAX 100LX network interface, configured in R_NETWORK_REC_CONFIG, performs the following:

- Communicates incoming data to the receiving FIFO of DMA channel 1
- Communicates status to DMA channel 1
- Checks the destination address of the incoming frame
- Checks the CRC of the incoming frame
- Checks the length of the incoming frame

For more information regarding DMA, please refer to chapter *7 DMA*.

9.5.1 Data Transfer to the Receiving FIFO

Incoming data is transferred, one byte at a time, to the receiving FIFO of DMA channel 1. If the destination address of the frame does not match the address in the network interface, or if there is an error in the frame, the frame is aborted and the remaining bytes in the frame are not sent to the FIFO. If an overrun occurs, the packet that caused the overrun is aborted, and reception continues with the next packet.

The frame sent to the FIFO contains the following information:

DA	Destination address (6 bytes)
SA	Source address (6 bytes)
L	Length or type (2 bytes)
Data	Frame data (+ pad if necessary) (46 to 1500 bytes)
CRC	Cyclic Redundancy Check (4 bytes)

The preamble and SOF fields are generated by the PHY/controller.

9.5.2 Address Recognition

The destination address (DA) field of the incoming frame is compared with the addresses set in the network interface. If the addresses do not match, the frame is aborted. Within the frame formats of the IEEE 802.3 standard, there are three types of destination addresses:

- Individual address (unicast)
- Group address (multicast)
- Broadcast (all nodes)

Individual address

The ETRAX 100LX can hold two individual addresses, MA0 and MA1, each 48 bits long. All 48 bits are compared in each address. This means that one or both of the individual addresses can be used to match a group address instead.

In R_NETWORK_REC_CONFIG, there is also one mode bit, ma0 and ma1 respectively, for each individual address to enable/disable recognition.

Group address

A 6-bit hash address is calculated from the 48 destination address (DA) bits:

Hash_address[5:0] =

DA[5:0] ^ DA[11:6] ^ DA[17:12] ^ DA[23:18] ^ DA[29:24] ^ DA[35:30] ^ DA[41:36] ^ DA[47:42]

Note 2: ^ = XOR

The hash address is used as an index for a 64-bit table (R_NETWORK_GA_0 and R_NETWORK_GA_1), which indicates whether or not to copy the frame. In R_NETWORK_REC_CONFIG, there is also the individual mode bit, which is used to select whether to only match group addresses (DA[47] == 1) or to also match individual addresses.

The different addresses above can also be used to get promiscuous mode. Set all bits in R_NETWORK_GA_0 and R_NETWORK_GA_1 to 1, and set the individual field of R_NETWORK_REC_CONFIG to receive (1).

Broadcast address

In R_NETWORK_REC_CONFIG, there is a mode bit broadcast to enable/disable the recognition of the broadcast address 0xFFFFFFFFFFFF.

9.5.3 Receiver CRC Check.

The incoming data is CRC checked according to the IEEE 802.3 standard. If an incorrect CRC is detected, the frame is aborted, and the CRC error counter crc_error or alignment error counter alignment_error is incremented in the register R_REC_COUNTERS. For the difference between these two error counters, please refer to the IEEE 802.3 standard.

With an MII interface, the CRC block also monitors the rxer input signal. If the rxer signal occurs during a reception, this is handled as if an incorrect CRC occurred.

The mode bit bad_crc in R_NETWORK_REC_CONFIG, enables/disables the abortion of frames due to incorrect CRC. If disabled, frames with CRC and alignment errors will also be received. Two status bits, crc_err and align_err, are transferred to DMA channel 1. DMA will put this status information in the status field of the DMA descriptor for the received frame.

9.5.4 Received Frame Length Check

The lengths of the incoming frames are checked against the specified limits of the IEEE 802.3 frame format:

- Minimum: 64 bytes
- Maximum: 1518 or 1522 bytes

The max length of a standard Ethernet packet is 1518 bytes. The length can be extended to 1522 bytes when VLAN tagging, according to IEEE 802.1q, is used. The max length is configured with the max_size field in R_NETWORK_REC_CONFIG.

If the frame length is outside the specified bounds, the frame is aborted. There are two mode bits oversize and undersize in R_NETWORK_REC_CONFIG to enable the minimum length and maximum length check separately.

9.6 Transmitter Logic Functions

The transmitter logic block, configured in R_NETWORK_TR_CTRL, performs the following functions:

- Adds preamble and start of frame delimiter to the beginning of the frame.
- Pads frame data with zeros if the length of the frame data falls below the required minimum of 46 bytes.
- Calculates and adds CRC to the end of the frame.
- Handles the access protocol including automatic retransmission (CSMA/CD or demand priority).

9.6.1 Transmission of Frames

Transmit data is read from the FIFO of DMA channel 0, one byte at a time. The frame from the FIFO should contain the following:

DA	Destination address (6 bytes)
SA	Source address (6 bytes)
L	Length or type (2 bytes)
Data	Frame data (0 to 1500 bytes) (Note 3)

Note 3: Minimum length of data is 46 bytes if automatic pad is not selected.

The transmitter adds the preamble and start-of-frame delimiter to the beginning of the frame, and the frame check sequence (CRC) to the end of the frame. There is an option to automatically add a pad before the CRC, if the data field of the frame is shorter than 46 bytes. The pad consists of all zeros. Automatic padding is selected by the pad bit in R_NETWORK_TR_CTRL.

With the MII interface it is possible to deliberately corrupt a frame by setting the tx_err bit (force network transmission error) in the DMA descriptor. The corruption of frames by the tx_err bit is only possible if the txer pin is configured for transmit error output and is connected to the txer pin of the transceiver.The automatic addition of CRC to the frame can be disabled by setting the crc mode bit in R_NETWORK_TR_CTRL. This allows corrupted frames to be sent, for example, for test purposes.

If the transmitter gets a FIFO empty status during transmission without also getting an eop, the transmission is stopped and the transmit underrun interrupt is activated. The transmitter remains stopped until the interrupt is cleared.

If underrun occurs the transmitter stops. R_DMA_CH0_FIRST points to the first descriptor in the packet that could not be sent.

9.6.2 CSMA/CD Access Protocol

The CSMA/CD access protocol is used in the IEEE 802.3 mode. The protocol listens continuously to carrier sense to see if the line is free. When a frame is queued for transmission, it is sent as soon as the line is free.

If a collision is detected during the transmission, the protocol tries to resend the frame 15 times (i.e. a total of 16 attempts are made). If the transmission is aborted due to excessive collisions, the excessive retry interrupt is issued, and the transmitter stops. The transmitter remains stopped until the interrupt is cleared.

If an excessive collision occurs, the transmitter stops. R_DMA_CH0_FIRST points to the first descriptor in the packet that could not be sent. When the interrupt is cleared, transmission is restarted.

Two mode bits in R_NETWORK_TR_CTRL, retry and cancel, modify the access protocol handling. They can be used for other standard modes, as well as for test purposes or to implement protocols that deviate from the standard.

Field	Description
retry	Retransmission can be disabled. The excessive_col interrupt will then be issued after the first collision.
cancel	A pending frame can be cancelled. This will inhibit any attempts to start sending a frame. If there is data in the FIFO but the transmission has not started, the frame will be aborted. If the transmission is already started it will be completed. No transmit retries will be made if the cancel bit is set. After the current frame is completed or aborted, the excessive_col interrupt will be issued.

9.6.3 Demand Priority Access Protocol

Demand priority access protocol is used in IEEE 802.12 mode. The ETRAX 100LX is designed to support this mode together with Texas Instruments TNETE211 (100-VG-AnyLAN2 PMD interface) or equivalents.

The Ethernet controller also supports the token ring frame format through an 802.3 MII interface with extended protocol, using a handshake protocol. Configuration for this feature is done in R_NETWORK_REC_CONFIG.

For more information about the operation in these modes please contact Axis Communications.

9.7 Management Interface

The MII management interface consists of two pins, mdio and mdc. These pins are controlled by software, and are configured in R_NETWORK_MGM_CTRL.

As an extension of the IEEE 802.3 management protocol, the mdio interrupt is also implemented on the mdio pin. An mdio interrupt will be issued if the mdio field in R_NETWORK_STAT is sampled low while the interrupt is enabled.

In SNI mode, the management interface for the ETRAX 100LX also includes the possibility to use MII_TXD[3:1] and TX_ERR as general outputs, and to use MII_RXD[3:1] and RX_ERR as general inputs.

9.8 Ethernet Error and Statistics Counters

The ETRAX 100LX contains the following Ethernet error and statistics counters which are read in R_REC_COUNTERS, R_TR_COUNTERS and R_PHY_COUNTERS:

Counter	Explanation
crc_error	Number of frames with crc errors
alignment_error	Number of frames with alignment errors
oversize	Number of oversized frames
congestion	Number of otherwise correct frames that were not received due to a FIFO full condition
single_col	Number of frames that were involved in exactly one collision
multiple_col	Number of frames that were involved in more than one collision
late_col	Number of frames that were involved in late collisions
deferred	Number of deferred transmit frames
carrier_loss	Number of transmit frames for which the carrier sense signal was not constantly present during the transmission
sqe_test_error	Number of transmitted frames for which the sqe test signal was not recognized

Each counter is 8 bits wide. The counters are cleared either when they are read or at system reset. When a counter reaches 255 it stops counting. If the network interface is disabled the counter values are preserved.

An interrupt (with the same name as the error counter) is generated for a counter when it reaches 128. Reading the counter will clear the associated interrupt. The interrupt status can be masked individually, but all counters share the same interrupt vector number (0x27).

9.9 Network interrupts

In addition to error and statistics counter interrupts, there are four network interrupts. Two of these interrupts are for the network receiver, one is for the network transmitter, and one is for the mdio pin. All have the internally generated vector number 0x26.

overrun

This interrupt is set when the network receiver experiences a FIFO overrun condition (congestion error). Two interrupts, congestion (See section 9.8) and overrun, are available, but usually only one of them should be enabled. The overrun interrupt should be used if software intervention is necessary when an overrun error occurs. The congestion error counter should be used if the only action needed is an error count.

This interrupt is cleared by reading the congestion field of R_REC_COUNTERS, an action which also clears the congestion interrupt.

underrun

This interrupt is set when the network transmitter experiences a FIFO underrun condition. This interrupt is cleared by setting the clr_error field in R_NETWORK_TR_CTRL.

excessive_col

This interrupt is set when the network transmitter experiences collisions for 16 consecutive transmission attempts. It is set after the first collision if the retry field in network interface register R_NETWORK_TR_CTRL is set to disable, and when the transmitter stops after the cancel field of R_NETWORK_TR_CTRL has been set. This interrupt is cleared by setting the clr_error field in R_NETWORK_TR_CTRL.

mdio

This interrupt is from the MII mdio pin. It is generated when the mdio pin is low. The interrupt should be masked off during normal data transfers over the mdio interface. This interrupt should be cleared in the external unit that is driving the MDIO pin.

For more information see chapter 17 *Interrupts.*

10 EIDE/ATA-2/ATA-3 INTERFACE

10.1 ATA Interface Pin Connection

The EIDE/ATA-2/ATA-3, or ATA interface for short, supports four ATA busses without external logic, see Figure 10-1 below.

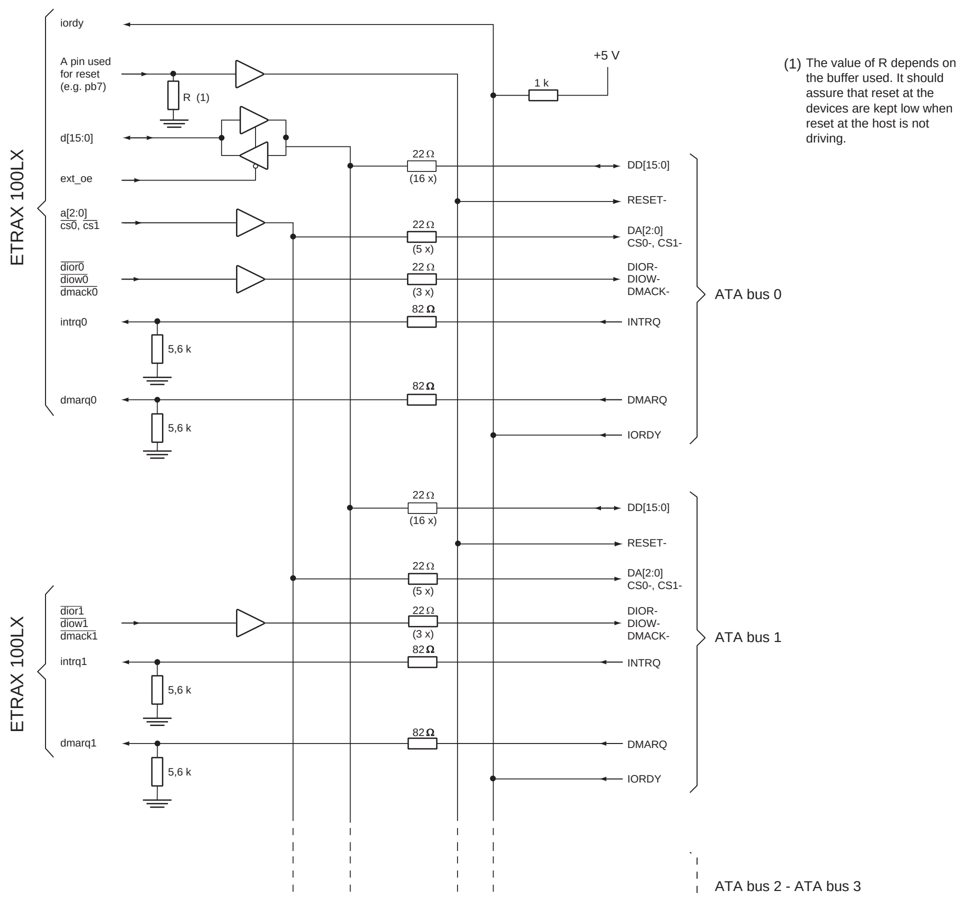


Figure 10-1 How to Connect the ATA Bus.

Each bus is capable of accessing two ATA devices, so up to a total of eight devices can be accessed by the ATA interface.

The ATA interface is enabled in the R_GEN_CONFIG register (See 18.11 *ATA Interface Registers*).

The reset signal of the ATA interface (RESET- (Device reset)) is not supported by the hardware of the ETRAX 100LX, but must be chosen from an available general I/O pin (e.g. General Port pa0 - pa7, General Port pb0 - pb7, g27, etc.) and handled by software. For more information regarding general I/O pins see chapter 19 *Electrical Information*.

10.2 EIDE/ATA-2/ATA-3 Interface Registers

An ATA/ATAPI device (e.g. a hard disk or CD-ROM drive) is controlled via a set of read/write registers in each device. By writing to these registers the ATA device can be made to perform commands such as reading or writing data to or from a disk. For more information about these registers please see the ATA-3 standard.

Communication between the ETRAX 100LX and ATA devices takes place via the following set of ETRAX 100LX registers:

Register	Function
R_ATA_CTRL_DATA	A 32-bit write only register to enable reading data from and writing data to ATA devices.
R_ATA_STATUS_DATA	A 32-bit read only register that indicates if the ATA interface is busy. It also indicates if the transmitter is ready, and holds data read from the ATA device.
R_ATA_CONFIG	A 32-bit write only register for enabling the ATA controller, DMA handshaking, and setup and hold time for register read/writes.
R_ATA_TRANSFER_CNT	A 32-bit read/write register used to set the number of bytes or 16-bit words transferred during DMA transfers.

For more detailed information about these registers, see chapter 18.11 *ATA Interface Registers*

10.3 Data Transfer

The ATA interface of ETRAX 100LX can be driven either by internal DMA or by using the register R_ATA_CTRL_DATA (see sections 10.3.3 ETRAX 100LX Register Access, and 10.3.4 ETRAX 100LX DMA Access below). Programmed input/output (PIO) is used for transferring commands to the ATA device, and for transferring data if the device does not support the DMA handshaking protocol.

Internal DMA of the ETRAX 100LX must not be confused with the DMA handshaking on the ATA bus that many ATA devices use. ETRAX 100LX DMA can be used both when directly accessing the ATA registers of the device Programmed Input/Output (PIO) and when using the DMA handshaking protocol of the device.

10.3.1 Programmed Input/Output (PIO)

Commands are always transferred to ATA devices using PIO, and the commands are written directly to the R_ATA_CTRL_DATA register in the ETRAX 100LX. Data can also be transferred to ATA devices in the same way. Data fetched from ATA devices is read in R_ATA_STATUS_DATA.

10.3.2 ATA DMA Handshaking

Most new ATA devices use DMA handshaking to transfer data. DMA handshaking is enabled in R_ATA_CONFIG.

The device sets its DMARQ signal high when it is ready to receive or deliver data. If the amount to be transferred to or from the device is large, it is possible that the device can neither accept nor produce all data in one burst. It will then lower its

DMARQ for periods of time during the transfer. It may take some time until the device is able to accept or produce more data after it has lowered its DMARQ, so this time can be used to talk to another ATA device. This is made possible through the ata_dmaend interrupt generated when the DMARQ signal goes low.

10.3.3 ETRAX 100LX Register Access

The R_ATA_CTRL_DATA register of the ETRAX 100LX is used for writing to or reading from the registers of ATA devices. Each time R_ATA_CTRL_DATA is written to, one transfer is made to or from the ATA device. The result (e.g. the data transferred from the ATA device) is read in R_ATA_STATUS_DATA.

10.3.4 ETRAX 100LX DMA Access

An alternative to register access is to let internal DMA of the ETRAX 100LX drive the ATA interface, which is then used to transfer data to and from the ATA device. Configuration to allow DMA to drive the ATA interface is done in R_GEN_CONFIG. ATA uses DMA channels 2 and 3.

A transfer counter, R_ATA_TRANSFER_CNT, is used. For each ATA transfer (8 or 16 bits) the counter is decremented. When the R_ATA_TRANSFER_CNT register reaches zero, the transfer is stopped. If data was transferred to the ETRAX 100LX, an end-of-packet (EOP) is signalled to ETRAX 100LX DMA.

10.4 Timing

If PIO is used, the time for transferring individual data varies between 140 ns and 600ns depending on what mode the ATA device is in (For details please see the ATA-3 standard).

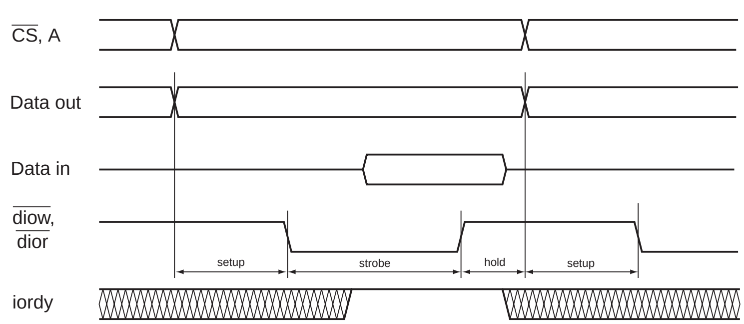


Figure 10-2 PIO Timing

When DMA handshaking is used, the time varies from 120 ns to 480 ns depending on the ATA mode.

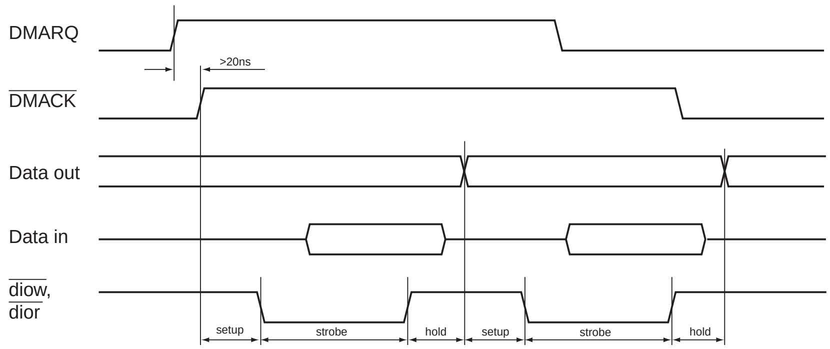


Figure 10-3 DMA Multiword (16-bit) Timing

Timing for both PIO and DMA handshaking is configured in R_ATA_CONFIG.

10.5 Interrupts

The ATA interface has nine interrupts:

ata_irq0, ata_irq1, ata_irq2, and ata_irq3

When ATA is in use, ata_irq0, ata_irq1, ata_irq2, or ata_irq3 is set when a unit on the ATA bus (0, 1, 2, 3 respectively) requests an interrupt. The interrupt is cleared through registers in the external unit on the ATA bus (0, 1, 2, 3 respectively).

ata_drq0, ata_drq1, ata_drq2, and ata_drq3

Whe ATA is in use, ata_drq0, ata_drq1, ata_drq2, or ata_drq3 is set when a unit on the ATA bus (0, 1, 2, 3 respectively) requests a DMA transfer. The interrupt is automatically cleared at the end of the DMA transfer on ATA bus (0, 1, 2, 3 respectively).

ata_dmaend

The ata_dmaend interrupt is set when the selected ATA unit releases its DMA request (transfer completed). The interrupt should be masked in R_IRQ_MASK0_SET except when an ATA DMA transfer has been started. It is automatically cleared when the next ATA DMA transfer commences.

11 ASYNCHRONOUS SERIAL PORTS

11.1 General

The ETRAX 100LX contains four complete asynchronous serial receivers/ transmitters with full buffering and parity control. Each asynchronous serial port has one handshake signal in each direction.

The receivers/transmitters support baud rates from 48 up to 1,843,200 baud, plus a non-standard baud rate of 6,250,000 baud.

11.2 Connection to Input/Output Pins

The pins of Asynchronous Serial Port p0 are available in all configurations. The other asynchronous serial port pins are multiplexed as shown in table 11-1 below:

Asynchronous Serial Port Pins	Multiplexed Pins
Asynchronous Serial Port p1	Synchronous Serial Port p1 USB Port p1 General I/O Port
Asynchronous Serial Port p2	SCSI-8 Port p0 SCSI-W Port ATA Port General I/O Port
Asynchronous Serial Port p3	Synchronous Serial Port p3 SCSI-8 Port p1 ATA Port General I/O Port

Table 11-1 Asynchronous serial port pin multiplexing

Pin configuration is done in registers R_GEN_CONFIG and R_GEN_CONFIG_II.

In order to set the correct default value of Asynchronous Serial ports p2 and p3, the hcfg pin should be tied to 0:

if (hcfg = 0), then {s0sel = 1, s1sel = 1}

For more information see section 19.11 *I/O Pin Default Values*.

11.3 Asynchronous Serial Port Registers

Table 11-2 below provides a summary of the asynchronous serial port registers. For more detailed information see chapter 18.8 *Serial Port Registers*.

Register	Function
R_SERIAL0_CTRL R_SERIAL1_CTRL R_SERIAL2_CTRL R_SERIAL3_CTRL	A 32-bit wide write only register for baud rate selection, serial transmitter/ receiver control, and serial data out.
R_SERIAL0_BAUD R_SERIAL1_BAUD R_SERIAL2_BAUD R_SERIAL3_BAUD	A byte wide write only register for transmitter/receiver baud rate selection.
R_SERIAL0_REC_CTRL R_SERIAL1_REC_CTRL R_SERIAL2_REC_CTRL R_SERIAL3_REC_CTRL	A byte wide write only register for serial receiver control.
R_SERIAL0_TR_CTRL R_SERIAL1_TR_CTRL R_SERIAL2_TR_CTRL R_SERIAL3_TR_CTRL	A byte wide write only register for serial transmitter control.
R_SERIAL0_TR_DATA R_SERIAL1_TR_DATA R_SERIAL2_TR_DATA R_SERIAL3_TR_DATA	A byte wide write only register for serial data out.
R_SERIAL0_READ R_SERIAL1_READ R_SERIAL2_READ R_SERIAL3_READ	A byte wide read only register for serial transmitter/receiver status, and serial data in.
R_SERIAL0_STATUS R_SERIAL1_STATUS R_SERIAL2_STATUS R_SERIAL3_STATUS	A byte wide read only register for serial transmitter/receiver status.
R_SERIAL0_REC_DATA R_SERIAL1_REC_DATA R_SERIAL2_REC_DATA R_SERIAL3_REC_DATA	A byte wide read only register for data in from the serial receiver.
R_SERIAL0_XOFF R_SERIAL1_XOFF R_SERIAL2_XOFF R_SERIAL3_XOFF	A 32-bit wide write only register for serial transmitter control, and xoff handling.
R_ALT_SER_BAUDRATE	A 32-bit wide write only register for alternative baud rate selection.
R_SERIAL_PRESCALE	A 16-bit write only register for the divide factor for serial clock prescaling.
R_SER_PRESC_STATUS	A 16-bit read only register for the current count value of the serial divide factor.

Table 11-2 Asynchronous serial port registers

11.4 Operation Modes

The serial port operation modes are configured in R_SERIAL*x*_CTRL. The receiver and transmitter can also be configured separately by using the registers R_SERIAL*x*_REC_CTRL and R_SERIAL*x*_TR_CTRL respectively. The following modes can be configured:

- Receiver and transmitter baud rate
- Odd, even, fixed or no parity
- 7 or 8 data bits

The transmitter can also be configured to send 1 or 2 stop bits, and to handle $\overline{\text{cts}}$ automatically or to ignore $\overline{\text{cts}}$. When automatic $\overline{\text{cts}}$ handling is selected, a high on the $\overline{\text{cts}}$ input will halt the transmitter after the ongoing byte, and keep it halted until $\overline{\text{cts}}$ becomes low again.

The $\overline{\text{rts}}$ output is also controlled by a bit in the R_SERIAL*x*_CTRL register.

In the R_SERIAL*x*_XOFF register, the serial port can be configured to stop transmission when the xoff character is received. The character code for xoff is also configurable.

Receiver and transmitter baud rate configuration can be done in either R_SERIAL*x*_REC_CTRL or R_SERIAL*x*_BAUD.

Asynchronous Serial Port p0 can be used when bootstrapping the ETRAX 100LX. See chapter 6 *Bootstrap Methods*.

Note 1: In order to avoid unnecessary repetition of port numbers, the *x*_ in the interrupt and register names stands for either 0_, 1_, 2_ or 3_, representing Asynchronous Serial Ports p0, p1, p2 and p3.

11.5 Baud Rate Selection

There are four different ways to set the baud rate, which is configured in R_ALT_SER_BAUDRATE:

- The default setting for R_ALT_SER_BAUDRATE is to choose one of the available predefined baud rates shown below in table 11-3. The predefined baud rate is set in R_SERIAL*x*_CTRL or R_SERIAL*x*_BAUD. The baud rate can be set individually for each port, and separately for input and output:

Predefined Baud rates		
300	9,600	230,400
600	19,200	460,800
1,200	38,400	921,600
2,400	57,600	1,843,200
4,800	115,200	6,250,000

Table 11-3 Predefined baud rates

- Use a scalable baud rate. The scalable baud rate can be set between 1,562,500 baud and 48 baud through a 16 bit divide factor loaded in R_SERIAL_PRESCALE. The resulting baud rate will be (3,125,000/ser_presc baud), where ser_presc is the divide factor in R_SERIAL_PRESCALE.
- Use an external baud rate clock which is enabled in R_GEN_CONFIG_II. The clock must be less than 20 MHz. The baud rate achieved from the external clock is the external clock divided by eight, so the maximum baud rate is 2,500,000 baud. The General I/O Port pb6 is assigned to receive the external baud rate clock.
- Use timer0 as the baud rate generator. The frequency generated with timer0, is configured in the R_TIMER_CTRL register. For transmission the timer output clock is used directly resulting in a maximum baud rate of 12.5 MHz. For receiving the output is divided by eight resulting in a maximum baud rate of 1,562,500 baud.
  For more detailed information see chapter 15 *Timers*.

11.6 CPU Controlled Operation

When the CPU controls serial port operation, the port can be polled or interrupt driven.

When the transmitter is ready to accept new data, the tr_ready field in R_SERIAL*x*_READ and R_SERIAL*x*_STATUS is set to ready, and the ser*x*_ready interrupt is generated. The tr_ready bit and the ser*x*_ready interrupt are cleared when data is written to the R_SERIAL*x*_TR_DATA register or to the data_out field of R_SERIAL*x*_CTRL.

When data is available from the serial receiver, the data_avail field in R_SERIAL*x*_READ and R_SERIAL*x*_STATUS is set to yes, and the ser*x*_data interrupt is generated. The data_avail bit and the ser*x*_data interrupt are cleared when data is read from the R_SERIAL*x*_REC_DATA register or from the data_in field of R_SERIAL*x*_READ.

If the serial receiver encounters a parity error, framing error or overrun error, the errors are indicated in the R_SERIAL*x*_READ and R_SERIAL*x*_STATUS registers. The error status is valid whenever the data_avail bit is set, and is cleared when data is read from the R_SERIAL*x*_REC_DATA register or from the data_in field of R_SERIAL*x*_READ.

11.7 DMA Controlled Operation

In R_GEN_CONFIG, the asynchronous serial ports can be connected to the internal DMA channels as shown in the table below:

Async Serial Port	DMA Channel	
	In	Out
Async Serial Port p0	Channel 7	Channel 6
Async Serial Port p1	Channel 9	Channel 8
Async Serial Port p2	Channel 3	Channel 2
Async Serial Port p3	Channel 5	Channel 4

In a DMA controlled operation, received data is entered into the FIFO of the connected internal DMA channel. The data in the FIFO will only be written out when it reaches the configured FIFO trip point (i.e. 16 or 32 bytes, which is configured in R_BUS_CONFIG). Therefore, after the last received byte, there may be up to 31 bytes left in the FIFO. The remaining FIFO contents can be written out to the memory by issuing an eop (end of packet) command to DMA. This is done by writing to the R_SET_EOP register.

In addition, the data_avail field in R_SERIAL*x*_READ and R_SERIAL*x*_STATUS is set when a data byte is received, and is cleared by a CPU read from the R_SERIAL*x*_REC_DATA register or from the data_in field of R_SERIAL*x*_READ. The same is also true for the ser*x*_data interrupt. Therefore, the data_avail field and/or the ser*x*_data interrupt can be used to detect when new data has been entered into the FIFO.

To handle receive errors in a DMA controlled operation, there are two different modes:

1 If the dma_err field in R_SERIAL*x*_CTRL is set to stop, a receive error generates an eop to DMA and stops the receiver so the erroneous byte is not entered into the FIFO. The error condition is cleared by a CPU read from the R_SERIAL*x*_REC_DATA register, or from the data_in field of R_SERIAL*x*_READ.

2 If the dma_err field is set to ignore, receive errors are ignored and the erroneous byte is entered in the FIFO.

During a DMA controlled transmission, DMA transfers may be temporarily stopped and CPU mode entered by disconnecting DMA in R_GEN_CONFIG. This can be used, for example, for inserting a CPU controlled xoff transmission into a DMA controlled data stream.

11.8 Asynchronous Serial Port Interrupts

Each asynchronous serial port has two interrupts, one for the receiver and one for the transmitter:

ser*x*_ready

This interrupt is set when the asynchronous serial port is ready to acquire new data for transmission. This interrupt is cleared when new data is written to the R_SERIAL*x*_TR_DATA register, or to the data_out field of R_SERIAL3_CTRL. The ser*x*_ready interrupt should be masked when DMA is used for data transfers.

ser*x*_data

This interrupt is set when input data is available on the port. It is cleared when data is read from the R_SERIAL*x*_REC_DATA register, or from the data_in field of R_SERIAL*x*_READ. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.

For more detailed information see chapter 17 *Interrupts.*

12 SYNCHRONOUS SERIAL INTERFACE

12.1 Overview

The synchronous serial interface (sync serial) in the ETRAX 100LX sends the transmission clock along with the data. Communication takes place between two parties: a master which generates the clock, and a slave. The ETRAX 100LX can be either the master or the slave.

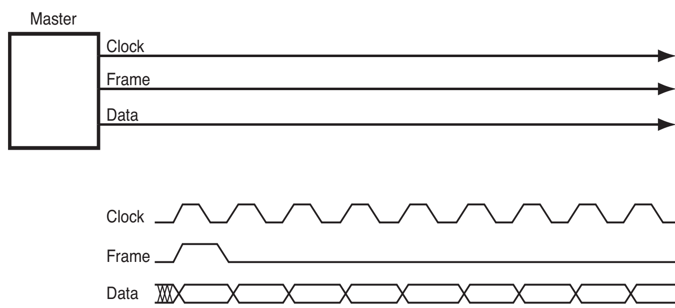


Figure 12-1 Simple Synchronous Serial Port

The sync serial interface has two synchronous serial ports. The first sync serial port is named *Synchronous Serial Port p1* and the other is *Synchronous Serial Port p3*. These two ports are multiplexed with Asynchronous Serial Ports p1 and p3 respectively, so Sync Serial Ports p0 and p2 do not exist.

12.2 Mode Selection

There are six different operation modes into which the sync serial ports of the ETRAX 100LX can be configured. Three different types of communication channels can be used, two of which are unidirectional, and one of which is bidirectional. Because there is one master and one slave in each type of channel, one synchronous serial port may be configured into one of the following six modes:

Mode	Description
Master Output	The ETRAX 100LX generates clock signals, and data is transferred to the external unit.
Master Input	The ETRAX 100LX generates clock signals, and data is transferred from the external unit.
Master Bidirectional	The ETRAX 100LX generates clock signals, and data is transferred in both directions.
Slave Output	The ETRAX 100LX sends data to the external unit depending on the incoming clock signals it receives.
Slave Input	The ETRAX 100LX receives data from the external unit depending on the incoming clock signals it receives.
Slave Bidirectional	The ETRAX 100LX sends and receives data from the external unit depending on the incoming clock signals it receives.

To properly configure the ETRAX 100LX, select the mode that is supported by the other circuit. If two ETRAX 100’s are connected to each other, any of the six modes are available.

12.3 Pin Usage

12.3.1 Pin Configuration

The sync serial ports are multiplexed onto the same pins as some other interface applications, see chapter *19 Electrical Information.*

To select sync serial port p1, the sermode1 bit in the R_GEN_CONFIG_II register must be set to sync (0x1). To select sync serial p3, the sermode3 bit in the R_GEN_CONFIG_II register must be set to sync (0x1), and the ser3 bit in the R_GEN_CONFIG register must be set to select (0x1).

For every configuration the sync serial interface has two output pins and two input pins, and the pin function is different depending on the mode of operation. To make all configurations complete, however, one extra bidirectional pin per port will be required. In order to use some synchronous serial port modes, 5 pins are needed.

Extra pins must be configured from General I/O Ports pb4 or pb7, which is done with the register R_PORT_PB_SET. For a detailed description of the different registers, please refer to chapter 18.18 *Synchronous Serial Port Registers.*

12.3.2 Pin Usage in the Different Modes

The signal names at the I/O pins differ, depending upon the mode in use. The following tables show the different I/O pin assignments of the two synchronous serial ports in each mode of operation.

Synchronous Serial Ports - Master Output Mode							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	-	J18	$\overline{\text{s1sel}}$	-	ss1_in1 ss3_in1	Not used by the synchronous serial ports in Master Output mode.
V19	$\overline{\text{cts1}}$	ss1status	J19	$\overline{\text{s1bsy}}$	ss3status	ss1_in2 ss3_in2	Serial busy input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1data	B19	$\overline{\text{s1en}}$	ss3data	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1frame	U16	pb7	ss3frame	ss1_io3 ss3_io3	Serial frame indicator output from respective port.

Synchronous Serial Ports - Master Input Mode							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1data	J18	$\overline{\text{s1sel}}$	ss3data	ss1_in1 ss3_in1	Serial data input to respective port.
V19	$\overline{\text{cts1}}$	ss1status	J19	$\overline{\text{s1bsy}}$	ss3status	ss1_in2 ss3_in2	Serial empty input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1frame	B19	$\overline{\text{s1en}}$	ss3frame	ss1_out2 ss3_out2	Serial frame indicator from respective port.
W17	pb4	-	U16	pb7	-	-	Not used by the synchronous serial ports in Master Input mode.

Synchronous Serial Ports - Slave Output Mode							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	ss1data	H20	s1sel	ss3data	ss1_out1 ss3_out1	Serial data output from respective port.
W20	$\overline{\text{rts1}}$	ss1status	B19	$\overline{\text{s1en}}$	ss3status	ss1_out2 ss3_out2	Serial empty output from respective port.
W17	pb4	-	U16	pb7	-	-	Not used by the synchronous serial ports in Slave Output mode.

Synchronous Serial Ports - Slave Input Mode

Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	-	H20	s1sel	-	ss1_out1 ss3_out1	Not used by the synchronous serial ports in Slave Input mode.
W20	$\overline{\text{rts1}}$	ss1status	B19	$\overline{\text{s1en}}$	ss3status	ss1_out2 ss3_out2	Serial busy output from respective port.
W17	pb4	ss1data	U16	pb7	ss3data	ss1_io3 ss3_io3	Serial data input to respective port.

Synchronous Serial Ports - Master Bidirectional Mode

Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1status	J18	$\overline{\text{s1sel}}$	ss3status	ss1_in1 ss3_in1	Serial busy input to respective port.
V19	$\overline{\text{cts1}}$	ss1idata	J19	$\overline{\text{s1bsy}}$	ss3idata	ss1_in2 ss3_in2	Serial data input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1odata	B19	$\overline{\text{s1en}}$	ss3odata	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1frame	U16	pb7	ss3frame	ss1_io3 ss3_io3	Serial frame indicator output from respective port.

Synchronous Serial Ports - Slave Bidirectional Mode							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	ss1status	H20	s1sel	ss3status	ss1_out1 ss3_out1	Serial busy output from respective port.
W20	$\overline{\text{rts1}}$	ss1odata	B19	$\overline{\text{s1en}}$	ss3odata	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1idata	U16	pb7	ss3idata	ss1_io3 ss3_io3	Serial data input to respective port.

12.4 Synchronous Serial Port Registers

The sync serial ports are served by two sets of dedicated registers, one set for each port. The table below summarizes the purpose of each register.

Registers	Function
R_SYNC_SERIAL_PRESCALE	A 32-bit write-only register that selects clock source and frame sync rate for both ports.
R_SYNC_SERIAL1_REC_DATA R_SYNC_SERIAL3_REC_DATA	32-bit read-only registers containing data in from the serial receivers.
R_SYNC_SERIAL1_REC_WORD R_SYNC_SERIAL3_REC_WORD	16-bit read-only registers containing data in from the serial receivers.
R_SYNC_SERIAL1_REC_BYTE R_SYNC_SERIAL3_REC_BYTE	8-bit read-only registers containing data in from the serial receivers.
R_SYNC_SERIAL1_STATUS R_SYNC_SERIAL3_STATUS	32-bit read-only registers containing status information from the ports.
R_SYNC_SERIAL1_TR_DATA R_SYNC_SERIAL3_TR_DATA	32-bit write-only registers containing data out to the serial receivers.
R_SYNC_SERIAL1_TR_WORD R_SYNC_SERIAL3_TR_WORD	16-bit write-only registers containing data out to the serial receivers.
R_SYNC_SERIAL1_TR_BYTE R_SYNC_SERIAL3_TR_BYTE	8-bit write-only registers containing data out to the serial receivers.
R_SYNC_SERIAL1_CTRL R_SYNC_SERIAL3_CTRL	32-bit write-only registers for configuration and control of each port.

For more detailed information about sync serial registers, refer to chapter 18.18 *Synchronous Serial Port Registers*.

12.5 Configuration

Besides the different modes described in section 12.2 *Mode Selection*, a number of things can be configured in the R_SYNC_SERIAL*x*_CTRL register:

- Word length can be either 8, 12, 16, 24 or 32 bits.
- A number of different frame synchronization modes can be configured.
- The ports can be controlled by the CPU or by DMA.
- Data can be sent with either the lsb or msb first.
- The active clock edge may be either positive or negative.
- All control signals can be individually inverted.
- The receiver and transmitter can be enabled and disabled individually as well.

12.6 Word Length

The sync serial interface supports a transmitted/received word length of either 8, 12, 16, 24 or 32 bits. The word length is configured with the wordsize field in R_SYNC_SERIAL*x*_CTRL.

Note 1: To avoid unnecessary repetition, the *x*_ in R_SYNC_SERIAL*x*_CTRL above and elsewhere in this chapter, stands for 1_ and 3_, representing Synchronous Serial Ports p1 and p3.

When the sync serial port is controlled by the CPU, transmit data is written to R_SYNC_SERIAL*x*_TR_DATA, and receive data is read from R_SYNC_SERIAL*x*_REC_DATA. Each read or write corresponds to one received or transmitted word of the selected word length.

The wordsize lsb bits are used by the interface. For word lengths other than 32-bits, the upper bits of R_SYNC_SERIAL*x*_TR_DATA are ignored by the transmitter, and the upper bits of R_SYNC_SERIAL*x*_REC_DATA contain invalid data.

The lower 8 bits or 16 bits of the transmit data can be written in the registers R_SYNC_SERIAL*x*_TR_BYTE and R_SYNC_SERIAL*x*_TR_WORD respectively. Each write to one of these registers will result in the transmission of one word of the selected word length. If the word length is wider than the written register, the remaining bits will be the same as in the previously transmitted word.

There are 8-bit and 16-bit registers for the data read as well, R_SYNC_SERIAL*x*_REC_BYTE and R_SYNC_SERIAL*x*_REC_WORD respectively.

When the sync serial interface is DMA driven, each transmitted/received word is represented by 1, 2 or 4 bytes in memory, depending on the selected word length:

Selected Word Length (bits)	Size In Memory (bytes)
8	1
12	2
16	2

Selected Word Length (bits)	Size In Memory (bytes)
24	4
32	4

12.7 Frame Synchronization

12.7.1 Frame Synchronization Modes

The following figures show the different types of frame synchronization configurations in R_SYNC_SERIAL*x*_CTRL:

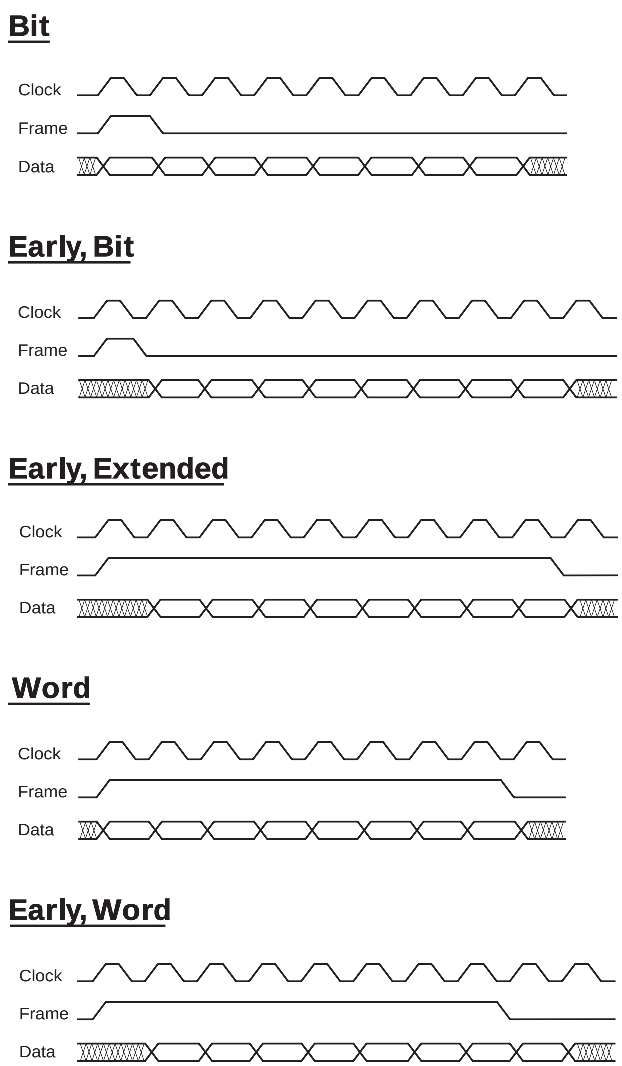


Figure 12-2 Synchronous Serial Type Configurations

Note 1: When the f_syncsize field in R_SYNC_SERIAL*x*_CTRL is set to *extended*, the field f_synctype must be set to *early*.

12.7.2 Frame Strobe Generation

When the sync serial interface operates in Master mode, the frame strobe output interval is controlled by the word_rate and frame_rate fields in R_SYNC_SERIAL_PRESCALE. The selected values are common to both ports.

The word_rate field selects the interval, in number of bit clocks, between the start of consecutive words transmitted or received. The word rate interval length is the value in the word_rate field + 1.

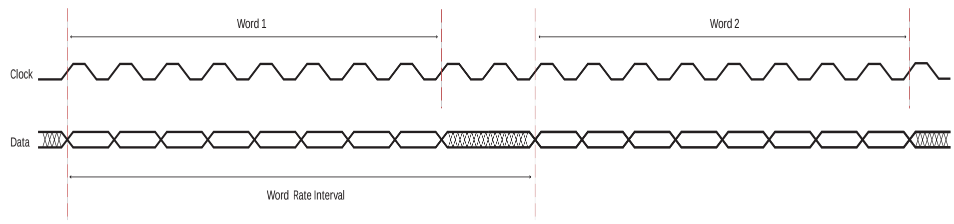


Figure 12-3 Word Rate Interval

The maximum value for the word rate interval is 1024 bits (i.e. word_rate field = 1023). The minimum value is restricted by the word length, see table 12-1 below:

Frame Sync Type: Bit or Word			Frame sync type: Early Bit, Early Word or Early Extended		
Word Length	Minimum word_rate Value	Minimum Word Rate Interval	Word Length	Minimum word_rate Value	Minimum Word Rate Interval
8	7	8	8	8	9
12	11	12	12	12	13
16	15	16	16	16	17
24	23	24	24	24	25
32	31	32	32	32	33

Table 12-1 Word length

The frame_rate field selects the number of words received or transmitted between each frame strobe output. The frame strobe interval is the value in the frame_rate field + 1. If the frame_rate field is set to 0, a frame strobe will be output for each word received or transmitted. This is the appropriate setting for most applications.

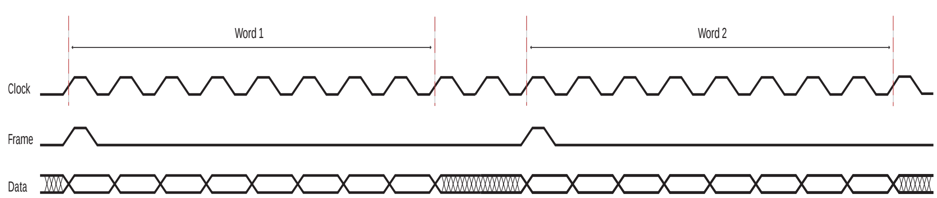


Figure 12-4 Frame Strobe Output with frame_rate Set to 0.

12.7.3 Stream Mode

It is possible to turn off frame synchronization, so that no frame strobe output is generated and the incoming frame strobe is ignored. Words will be transmitted or received back-to-back regardless of the word rate setting. This mode is used if the transferred data contains embedded synchronization.

12.8 Clocking

12.8.1 Clock Generator

There are three possible clock sources for the sync serial interface:

- Internally generated codec clock.
- Externally generated codec clock.
- Baudrate clock from the async serial port.

Clock selection is made in register R_SYNC_SERIAL_PRESCALE. Fields clk_sel_u3 and clk_sel_u1 are used to select either the codec clock or baudrate clock. The source for the codec clock (internal or external) is defined when *operation mode* is selected.

Internally Generated Codec Clock

The base clock is 4.096 MHz.

For selectable clock division, the base clock is then divided using a prescaler with a division factor of 1 to 128 before it is used by the sync serial port. The division factor is set in the prescaler field. The selected value is common to both ports.

Baudrate Clock from Async Serial Ports

When this mode is used, the clock is generated from the baud rate clock generator which is used in the async serial ports. This clock generator may be configured in a number of different ways, which is selected in the R_ALT_SER_BAUDRATE register and in the tr_baud field of R_SYNC_SERIAL*x*_CTRL. For a description of the different possibilities, refer to chapter 11.5 *Baud Rate Selection*.

Before this clock is used, the frequency is divided by two. Thus, the selectable frequency settings in the tr_baud field of the R_SYNC_SERIAL*x*_CTRL register, are half of the selected frequency for the asynchronous serial ports.

For the baudrate selection modes that do not use the tr_baud field, the following formula applies:

$$\text{sync serial bit clock} = \frac{\text{async serial baud rate}}{2}$$

12.8.2 Data Sampling

Sampling of incoming signals is controlled by the clk_polarity field in R_SYNC_SERIAL*x*_CTRL. The outgoing clock may be inverted with the clk_driver field.

When clocks are not inverted, output signals are changed on the rising edge, and input signals are sampled on the falling edge of the clock, see figure 12-5 below.

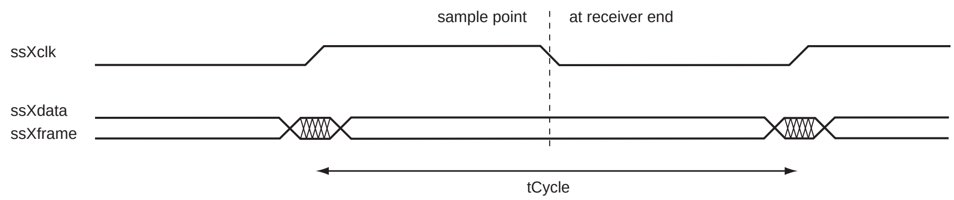


Figure 12-5 Master Output/Slave Input

12.8.3 Clock Gating

When clock gating is turned on (clk_mode is set to gated (1)), every negative clock edge is a bit strobe. When nothing is sent, the clock is held low. However, if clock gating is turned off (clk_mode is set to normal (0)), the clock will run continuously. Figure 12-6 below shows an example of the input mode with clock gating on:

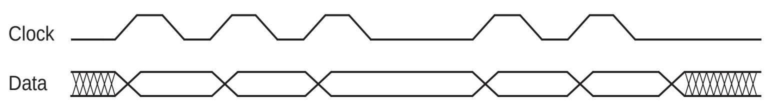


Figure 12-6 Input Mode with Clock Gating On

Note that clock gating does not work correctly with flow_ctrl set to enable (1) because the transmitter will stop clocking and never sample the status signal after its initial rise.

If more than one word is transferred between frames, as in figure 12-7 below, the clock must be gated for the receiver to understand the incoming data because the word strobe signal is not sent (only the frame strobe).

With clock gating turned on, the clock will only be active when the information is sent. In this case, every active clock edge contains information, and as a result, many words per frame can be understood by the receiver.

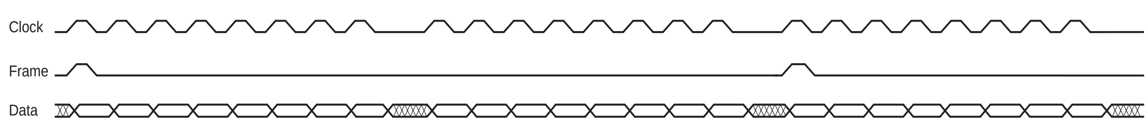


Figure 12-7 Output Mode with Clock Gating on, 2 Words Per Frame

The number of words per frame is selected in R_SYNC_SERIAL*x*_PRESCALE.

12.9 Flow Control

In the R_SYNC_SERIAL*x*_CTRL register, if the flow_ctrl field is set to off (0), the amount of transferred data is defined by the selected word rate. Data must be available when they are to be sent, and storage must be available before more data is received. If this is not the case, the failing unit will go into an error state which may be detected by reading the status register: R_SYNC_SERIAL*x*_STATUS.

When flow_ctrl is set to on (1), one or more status signals between units are used to pause the transfer until the receiver is prepared. This will delay word and frame synchronization. The master unit is paused by the slave if the slave can not deliver or receive data. If the master runs out of buffers, the start frame/word is delayed internally until the congestion is cleared. An overrun/underflow error will be reported if the transmitter or receiver over/under run their FIFO's.

When the error field in R_SYNC_SERIAL*x*_CTRL is set to ignore (1), the transmission is continued after next frame sync even if an error is detected. When error is set to normal (0), the transfer is halted when an error is detected. The status signal should be seen as the way the slave halts the transfer if data/storage is not available, and the frame signals the way the master controls the same variables.

The frame pulse, represented by a dotted line, is skipped.

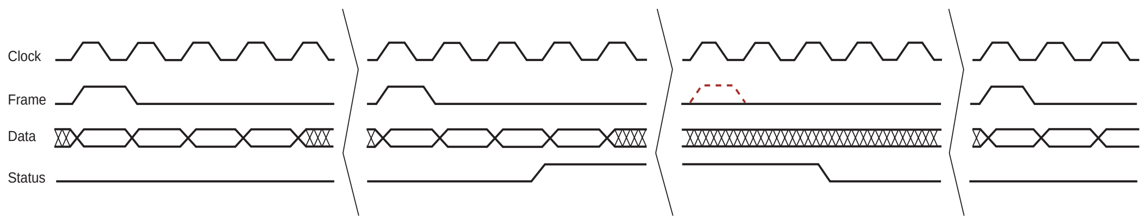


Figure 12-8 Flow Control

12.10 Interrupts

There are two interrupts for each sync serial port. The same interrupt bits in the R_IRQ_MASK1_RD register that are used for the asynchronous serial ports are also used for the sync serial interface: fields ser1_ready, ser1_data, ser3_ready, and ser3_data. For detailed information see chapter *18 Interrupts*.

The ser1_ready or ser3_ready interrupt is generated when the sync serial port is ready to accept a new data word to be transmitted, i.e. whenever the tr_ready field of the R_SYNC_SERIAL*x*_STATUS register is set. The interrupt is cleared when new data is written to one of the port data transmit registers R_SYNC_SERIAL*x*_TR_DATA, R_SYNC_SERIAL*x*_TR_WORD or R_SYNC_SERIAL*x*_TR_BYTE.

The ser1_data or ser3_data interrupt is generated when the sync serial port has received a new data word, i.e. whenever the data_avail field of the R_SYNC_SERIAL*x*_STATUS register is set. The interrupt is cleared when the data is read from one of the port data receive registers R_SYNC_SERIAL*x*_REC_DATA, R_SYNC_SERIAL*x*_REC_WORD or R_SYNC_SERIAL*x*_REC_BYTE.

12.11 Using The Sync Serial Ports with DMA

The sync serial ports can be driven by the CPU or by DMA. Sync serial port 1 uses internal DMA channel 8 for output, and DMA channel 9 for input. Port 3 uses DMA channel 6 for output and DMA channel 7 for input.

When DMA is used, the dma_enable field in R_SYNC_SERIAL1_CTRL register must be set, and the ports must be connected to their respective DMA channels, which is done in the R_GEN_CONFIG register.

For input transfers with DMA, the incoming data will be stored in the fifo of the DMA channel, and will not be written out to memory until the fifo is filled with 16 or 32 bytes (depending on the DMA burst length set in the R_BUS_CONFIG register). The DMA can be forced to write out the fifo contents to memory, by setting the appropriate bits in the R_SET_EOP register. The DMA will then set an eop in the current DMA descriptor, and advance to the next descriptor.

For further information on DMA operation, see chapter *7 DMA*.

13 PARALLEL PORTS

ETRAX 100LX has two parallel ports for the connection of high-speed peripheral devices. These ports are designated p0 and p1 respectively and the characteristics and operating principles of both ports are similar.

The parallel ports can be used through register access or internal direct memory access (DMA). The ports can be configured to communicate with compatible peripherals by using various protocols, including:

- IBM XT/AT compatible Centronics;
- IBM PS/2 compatible Centronics;
- Hewlett Packard Fast Mode;
- Fastbyte protocol;
- IEEE 1284 Byte, Nibble, Extended Capability Port (ECP), and Enhanced Parallel Port (EPP) modes;
- ECP wide (16-bit) mode.

A port designated parallel port-W is available when ETRAX 100LX operates in the ECP wide (16-bit) mode. Parallel port-W uses the control and status signals of port p0, together with the 8-bit data lines of p0 and p1 to form a word-wide data bus.

Inputs and outputs to and from the parallel ports are multiplexed on to the same pins as some other interface applications (See chapter 19.10 *Multiplexed Interfaces*).

13.1 Parallel Port Registers

The parallel ports are served by two sets of dedicated registers, one for each port. The required mode of operation and all operational parameters are established, written and read in these registers. The table below summarises the purpose of each register.

Registers	Function
R_PAR0_CTRL_DATA R_PAR1_CTRL_DATA	32-bit write-only registers containing discrete control bits and data for the respective port.
R_PAR0_CTRL R_PAR1_CTRL	8-bit write-only registers for the selection of port control signals oe, seli, autofd, strb and init.
R_PAR0_STATUS_DATA R_PAR1_STATUS_DATA	32-bit read-only registers containing status bits and data for the respective port.
R_PAR0_STATUS R_PAR1_STATUS	16-bit read-only registers containing status bits for the respective port.
R_PAR0_CONFIG R_PAR1_CONFIG	32-bit write-only registers for configuration and control of each port.
R_PAR0_DELAY R_PAR1_DELAY	32-bit write-only registers in which data transfer timing periods are set individually for each port.
R_PAR_ECP16_DATA	In ECP wide (16_bit) mode, this read/write register contains the last data word sent or received at parallel port p0.

For more detailed information on the parallel port registers, please refer to chapter 18.10 *Parallel Port Registers*.

13.2 Modes of Operation

The modes of operation supported by ETRAX 100LX are:

- IEEE 1284 Compatibility (Centronics) mode;
- IBM Fastbyte;
- IEEE-1284 Nibble mode;
- IEEE-1284 Byte mode;
- IEEE-1284 ECP mode;
- ECP wide (16-bit) mode (parallel port-W);
- IEEE 1284 EPP mode;
- Manual mode - this is the default mode of ETRAX 100LX - see the note below.

Note: 1 To switch to any other mode within an IEEE 1284 transaction, a negotiation phase is necessary. The negotiation phase, as well as termination and host recovery, are not supported in ETRAX 100LX and therefore it must be set up by software in the Manual mode.

13.2.1 IEEE-1284 Compatibility (Centronics) Mode

The IEEE-1284 Compatibility (Centronics) mode is a simplex mode for forward data transfers (ETRAX 100LX to peripheral).

The significant signals used by the parallel ports in Compatibility (Centronics) mode, and the corresponding signal names at the multiplexed I/O interface, are listed in the table below.

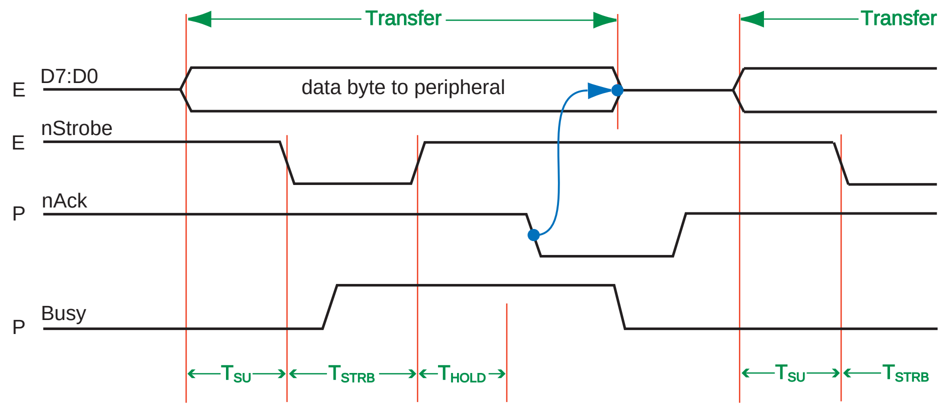
Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	Port_1			
p0d7 - p0d0	p1d7 - p1d0	D7:D0	Data byte from port to peripheral.	ETRAX 100LX
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	nStrobe	Strobe signal. Set low to initiate a data transmission.	ETRAX 100LX
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	nAck	Handshake line. Set low to indicate that the peripheral is ready to receive data.	Peripheral
p0busy	p1busy	Busy	Handshake line. Set high to indicate that the peripheral is busy and thus unable to receive data.	Peripheral

In the Compatibility (Centronics) mode only, ETRAX 100LX can be configured for different methods of handshaking with acknowledgment signal nAck. Each port's response to nAck is established by oe_ack (bit 3) or ign_ack (bit 4) in the port's configuration register R_PAR0_CONFIG or R_PAR1_CONFIG respectively.

Note: 2 The oe_ack bit and the ign_ack bit cannot both be asserted (set to 1), otherwise a conflict will occur.

Bit oe_ack asserted

When a data byte is ready, ETRAX 100LX checks that the Busy signal is not asserted, confirming that the peripheral can receive data. When setup time T_{SU} elapses, ETRAX 100LX asserts nStrobe and the transfer of the data byte commences. The nStrobe signal remains low for the duration of T_{STRB}, then the low/high transition of nStrobe initiates T_{HOLD}. The data transfer continues until T_{HOLD} elapses or until the peripheral asserts the nAck signal, whichever occurs last. Acknowledgment from the peripheral is thus necessary before ETRAX 100LX can place a new data byte on the bus.



T_{SU} can be set between 10 ns and 5 µs in steps of 20 ns.

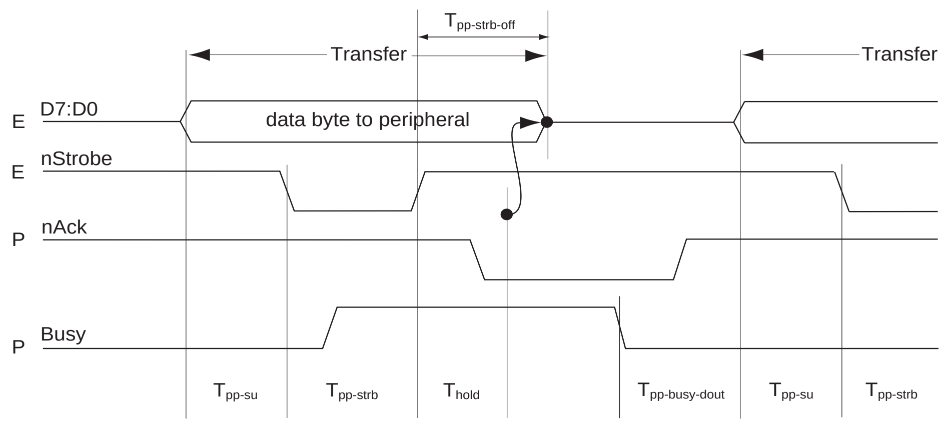
T_{STRB} and T_{HOLD} can be set between 20 ns and 5 µs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved arrow represents a sequential dependency.

E = ETRAX 100LX (host)   P = peripheral

Figure 13-1   Compatibility (Centronics) Mode Timing with nAck Succeeding T_{HOLD}



T_{su} can be set between 10 ns and 5 µs in steps of 20 ns.

T_{strb} and T_{hold} can be set between 20 ns and 5 us in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

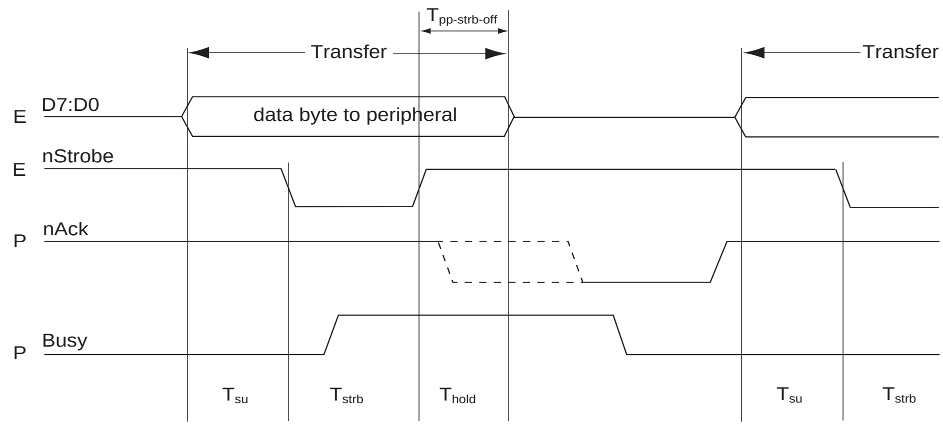
E = ETRAX 100LX (host)   P = Peripheral

Figure 13-2   Compatibility (Centronics) Mode Timing with nAck Preceding T_{HOLD}

Bit oe_ack deasserted

When a data byte is ready, ETRAX 100LX checks the Busy signal to confirm that the peripheral can receive data. When T_{SU} elapses, signal nStrobe is asserted by ETRAX 100LX to start the transfer of the data byte. The nStrobe signal remains low for the duration of T_{STRB}, then the rising edge of nStrobe initiates T_{HOLD}. The data transfer continues until T_{HOLD} elapses, irrespective of the state of the nAck signal.

In this configuration, ETRAX 100LX waits for an acknowledgment from the peripheral before placing the next data byte on the bus. The low/high transition of the nAck signal is necessary to trigger the next assertion of nStrobe.



T_{su} can be set between 10 ns and 5 us in steps of 20 ns.

T_{strb} and T_{hold} can be set between 20 ns and 5 us in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

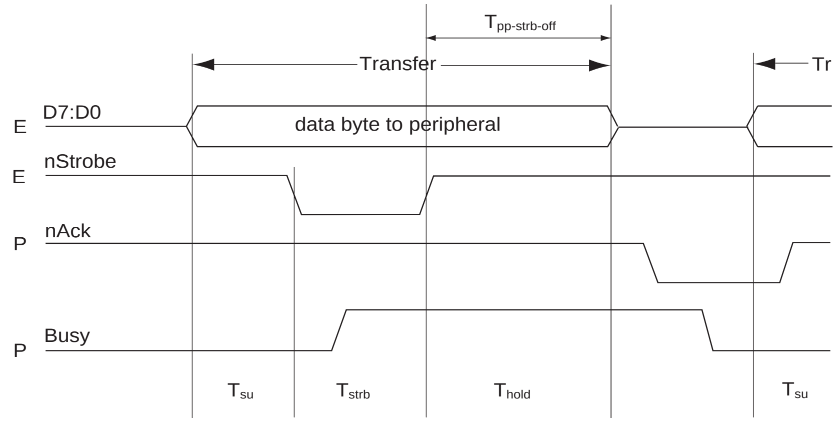
E = ETRAX 100LX (host)   P = Peripheral

Figure 13-3   Compatibility (Centronics) Mode Timing with Byte Transfers Terminated by T_{HOLD}

For this example to work, the ign_ack bit must be set to 0 (wait).

Bit ign_ack asserted

If the port is configured to ignore the nAck signal, only the Busy signal is monitored by ETRAX 100LX. When a data byte is ready, ETRAX 100LX checks the Busy signal to confirm that the peripheral can receive data. When setup time T_{SU} elapses, ETRAX 100LX asserts nStrobe and the data transfer commences. The nStrobe signal remains low for the duration of T_{STRB}, then its rising edge initiates T_{HOLD}. The data transfer stops when T_{HOLD} elapses.



T_{su} can be set between 10 ns and 5 us in steps of 20 ns.

T_{strb} and T_{hold} can be set between 20 ns and 5 us in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

E = ETRAX 100LX (host) P = Peripheral

Figure 13-4 Compatibility (Centronics) Mode Timing - Ignore nAck

Compatibility (Centronics) mode time periods

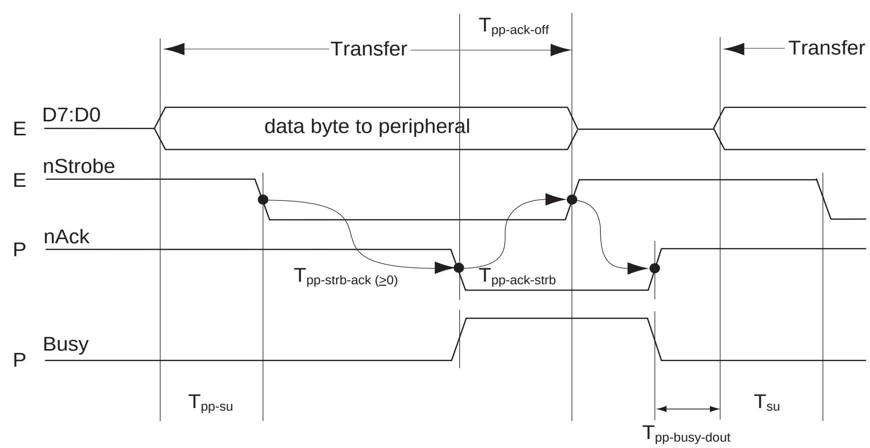
The data setup time (T_{SU}), the strobe time (T_{STRB}), and the data hold time (T_{HOLD}) are set individually for each port in register R_PAR0_DELAY or R_PAR1_DELAY respectively.

13.2.2 Fastbyte Mode

Fastbyte is a simplex mode for forward data transfer (ETRAX 100LX to peripheral) with four-phase handshaking. The significant signals used by the parallel ports in Fastbyte mode, and the corresponding signal names at the multiplexed I/O interface, are listed in the table below.

Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	**Port_1**			
p0d7 - p0d0	p1d7 - p1d0	D7:D0	Data byte from port to peripheral.	ETRAX 100LX
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	nStrobe	Strobe signal. Set low to initiate a data transmission.	ETRAX 100LX
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	nAck	Handshake line. Set low to indicate that the peripheral is ready to receive data.	Peripheral
p0busy	p1busy	Busy	Handshake line. Set high to indicate that the peripheral is busy and thus unable to receive data.	Peripheral

When data is ready ETRAX 100LX checks that the Busy signal is not asserted, confirming that the peripheral can receive data. After setup time T_{SU} has elapsed, ETRAX 100LX asserts the nStrobe signal. When the peripheral asserts nAck, ETRAX 100LX de-asserts nStrobe and the data transfer ceases.



T_{su} can be set between 10 ns and 5 µs in steps of 20 ns.

The curved lines represent sequential dependencies.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

E = ETRAX 100LX (host)   P = Peripheral

Figure 13-5 Fastbyte Mode Timing

Fastbyte mode time periods

For both ports, setup time T_{SU} is configurable in the respective internal register R_PAR0_DELAY or R_PAR1_DELAY (See chapter 18.10 *Parallel Port Registers*).

13.2.3 IEEE-1284 Nibble Mode

This is a simplex mode for reverse data transfers only (peripheral to ETRAX 100LX). The significant signals used by the parallel ports in Nibble mode, and the corresponding signal names at the multiplexed I/O interface, are listed in the table below.

Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	Port_1			
$\overline{\text{p0autofd}}$	$\overline{\text{p1autofd}}$	HostBusy	Handshake line. Set low to indicate that the respective port is ready to receive data, and high to indicate that the port has received a nibble.	ETRAX 100LX
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	PtrClk	Handshake line. Set low to indicate a valid nibble, and high to acknowledge that the respective port has received a nibble.	Peripheral
$\overline{\text{p0fault}}$	$\overline{\text{p1fault}}$	nDataAvail	Used for data bit 0, then 4.	Peripheral
p0select	p1select	XFlag	Used for data bit 1, then 5.	Peripheral
p0perror	p1perror	AckDataReq	Used for data bit 2, then 6.	Peripheral
p0busy	p1busy	PtrBusy	Used for data bit 3, then 7.	Peripheral

A peripheral requests attention from ETRAX 100LX with the initial handshaking between signals PtrClk, HostBusy and AckDataReq, which generates an interrupt to the respective parallel port (par0_peri for port p0, par1_peri for port p1). ETRAX 100LX responds to the interrupt when it is ready to receive data (See section 13.3 *Parallel Port Interrupts*).

The transfer of data takes place on the status signal lines in two cycles, one nibble at a time. Bits 0 to 3 are sent first and bits 4 to 7 follow after a handshake between PtrClk and HostBusy. The data bus D7:D0 and the nStrobe signal are not used in IEEE 1284 Nibble mode.

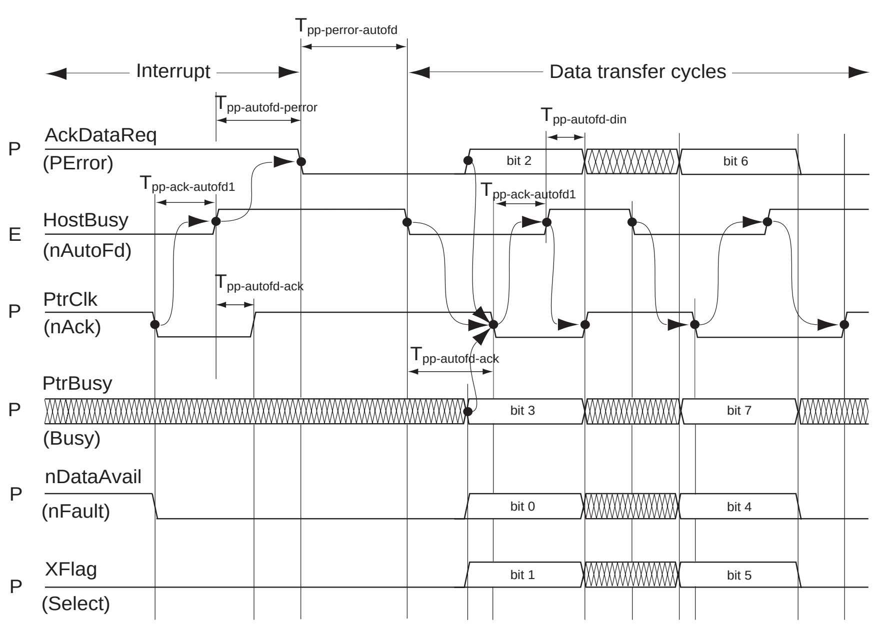


Figure 13-6 Nibble Mode Timing

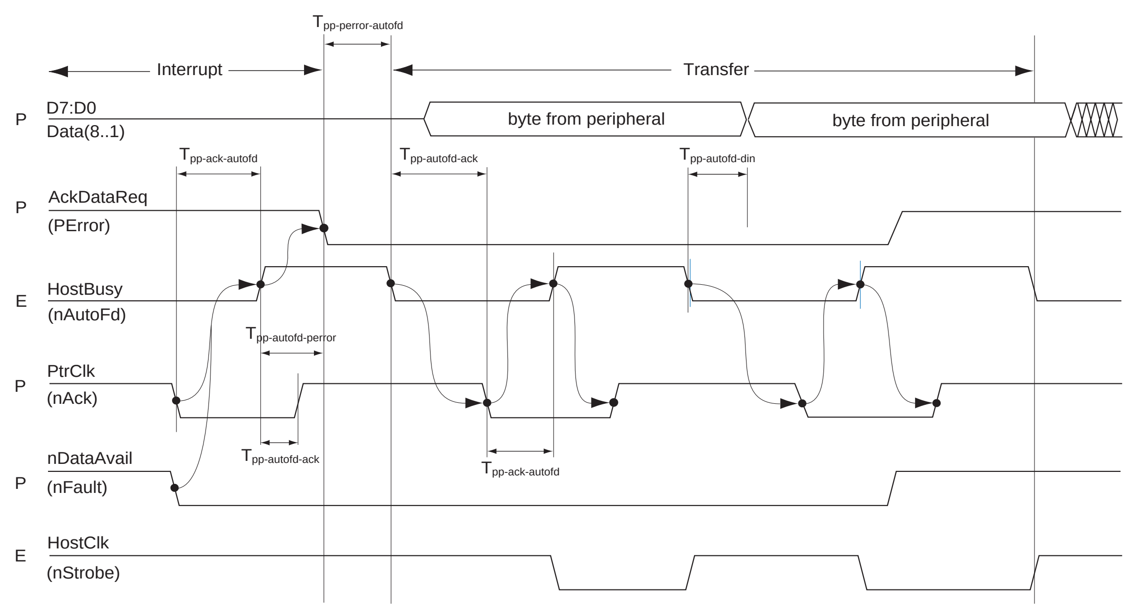
13.2.4 IEEE-1284 Byte Mode

This is a simplex mode for reverse data transfers (peripheral to ETRAX 100LX). The procedure is similar to the IEEE-1284 Nibble mode, the difference being that an entire byte is transferred simultaneously on the data bus.

The significant signals used by the parallel ports in this mode, and the corresponding signal names at the multiplexed I/O interface, are listed in the table below.

Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	Port_1			
p0d7 - p0d0	p1d7 - p1d0	D7:D0	Data bytes from peripheral to the respective port.	Peripheral
p0perror	p1perror	AckDataReq	Set low to acknowledge that the respective port is ready.	Peripheral
$\overline{\text{p0autofd}}$	$\overline{\text{p1autofd}}$	HostBusy	Handshake line. Set low to indicate that the respective port is ready to receive a byte, and high to indicate that the port has received the byte.	ETRAX 100LX
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	HostClk	Set low at the end of each byte transfer to indicate that the respective port has received a byte.	ETRAX 100LX
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	PtrClk	Handshake line. Set low to indicate a valid byte on the data lines, and high to acknowledge that the respective port has received a byte.	Peripheral
$\overline{\text{p0fault}}$	$\overline{\text{p1fault}}$	nDataAvail	Set low by peripheral to indicate that data is available.	Peripheral

Handshaking is realized by the $\overline{\text{PtrClk}}$ signal controlled by the peripheral, and the HostBusy signal controlled by ETRAX 100LX. Signal HostClk is pulsed low by ETRAX 100LX at the end of each byte transfer to indicate that the byte has been received.



The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host) P = Peripheral

Figure 13-7 IEE-1284 Byte Mode Timing

13.2.5 IEEE-1284 ECP Mode (Forward and Reverse)

The ECP mode supports half-duplex (forward and reverse), parallel data exchange between ETRAX 100LX and a peripheral, at transfer rates of up to 6 Mbyte/s. The protocol provides for separate data and command cycles, and two types of command cycle:

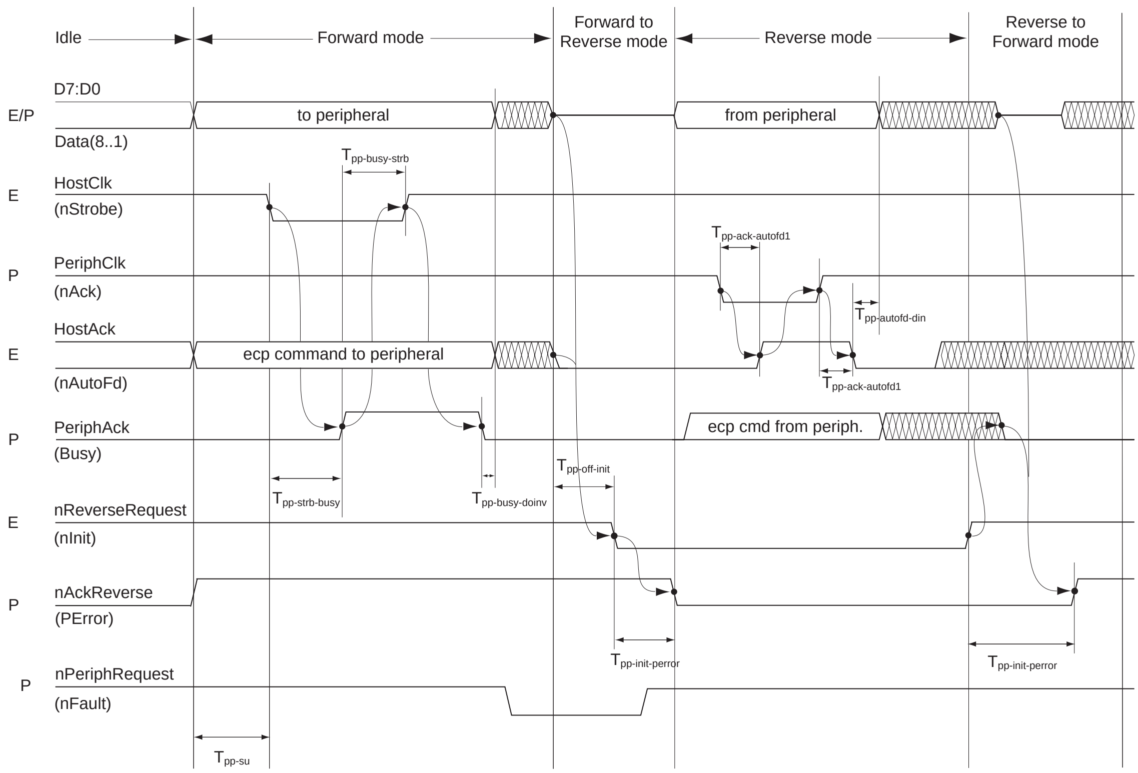
- data compression by Run Length Encoding (RLE) implemented in hardware;
- channel addressing, which must be handled in software.

The significant signals used by the parallel ports in the 1284 ECP mode, and the corresponding signal names at the multiplexed I/O interface, are listed in the table below.

Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	**Port_1**			
p0d7 - p0d0	p1d7 - p1d0	D7:D0	Carries data bi-directionally (half-duplex).	Both
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	HostClk	Handshakes with PeriphAck to transfer data or addresses in the forward direction. Set low to indicate valid data, and set high to clock data/addresses to the peripheral.	ETRAX 100LX
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	PeriphClk	Handshakes with HostAck to transfer data or addresses in the reverse direction.	Peripheral
$\overline{\text{p0autofd}}$	$\overline{\text{p1autofd}}$	HostAck	Provides command/data status in the forward direction. Handshakes with PeriphClk to transfer data in the reverse direction. Set high to indicate a data cycle.	ETRAX 100LX
p0busy	p1busy	PeriphAck	Handshakes with HostClk to transfer data or addresses in the forward direction. Provides command/data status in the reverse direction.	Peripheral
$\overline{\text{p0init}}$	$\overline{\text{p1init}}$	nReverseRequest	Set low to switch channel to reverse direction.	ETRAX 100LX
p0perror	p1perror	nAckReverse	Set low to acknowledge nReverseRequest.	Peripheral
$\overline{\text{p0fault}}$	$\overline{\text{p1fault}}$	nPeriphRequest	Set low to indicate that reverse data is available.	Peripheral

The HostAck signal is used to distinguish between data and command cycles. When HostAck is asserted, a data cycle is taking place: when HostAck is low a command cycle is taking place. During a command cycle, bit 7 of the data byte is used to indicate RLE or channel addressing. When bit 7 is set to 0, a run length count is being sent: when bit 7 is set to 1, a channel address is being sent.

The timing diagram below shows the transfer of data and commands in the forward direction, followed by a transfer of data and commands in the reverse direction.



T_{SU} can be set between 10 ns and 5 µs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host) P = Peripheral

Figure 13-8 ECP Mode Timing

ECP mode time periods

For both ports, setup time T_{SU} is configurable in the respective internal register R_PAR0_DELAY or R_PAR1_DELAY.

Stall detection

In ECP mode, a stall condition occurs if the peripheral is unable to accept a data byte sent by ETRAX 100LX. In this event ETRAX 100LX can abort the data transfer by asserting signal nReverseRequest. Regardless of whether the peripheral has already accepted the byte, it discards the byte and acknowledges ETRAX 100LX by asserting nAckReverse. ETRAX 100LX responds by de-asserting nReverseRequest.

Stall detection can be implemented in software by means of the tr_rdy bit (17), in register R_PAR0_STATUS_DATA or R_PAR1_STATUS_DATA. This bit is controlled exclusively by the respective parallel port. It is set to 1 when the port reads a new data byte, and to 0 when the software writes to R_PAR0_CTRL_DATA (7:0) or R_PAR1_CTRL_DATA (7:0). The stall condition can therefore be detected by writing to the control and data register, wait for a period to elapse (e.g. one second), and then read the status of tr_rdy. If this bit is set to 0, then data has not been sent.

An example of the code necessary to implement stall detection in ECP mode when using DMA is:

```
while (dma transfer is not ready) {
  WRITE to R_PARn_CTRL_DATA.data
  WAIT (e.g. 1s)
  READ R_PARn_STATUS_DATA.tr_rdy
  if (tr_rdy==0) {
    stall
  }
}
```

A similar work-around can be devised to implement stall detection in ECP mode when using registers.

13.2.6 ECP Wide (16-Bit) Mode

The ECP wide (16-bit) mode is not an IEEE 1284 mode. It is a feature of ETRAX 100LX that is almost identical to the IEEE 1284 ECP mode, except that data words are transferred instead of bytes. ECP wide (16-bit) mode is implemented in parallel port p0 only. Run Length Encoding (RLE) is not supported in this mode. The significant signals used by parallel port-W, and the corresponding signal names at the multiplexed I/O interface, are listed below.

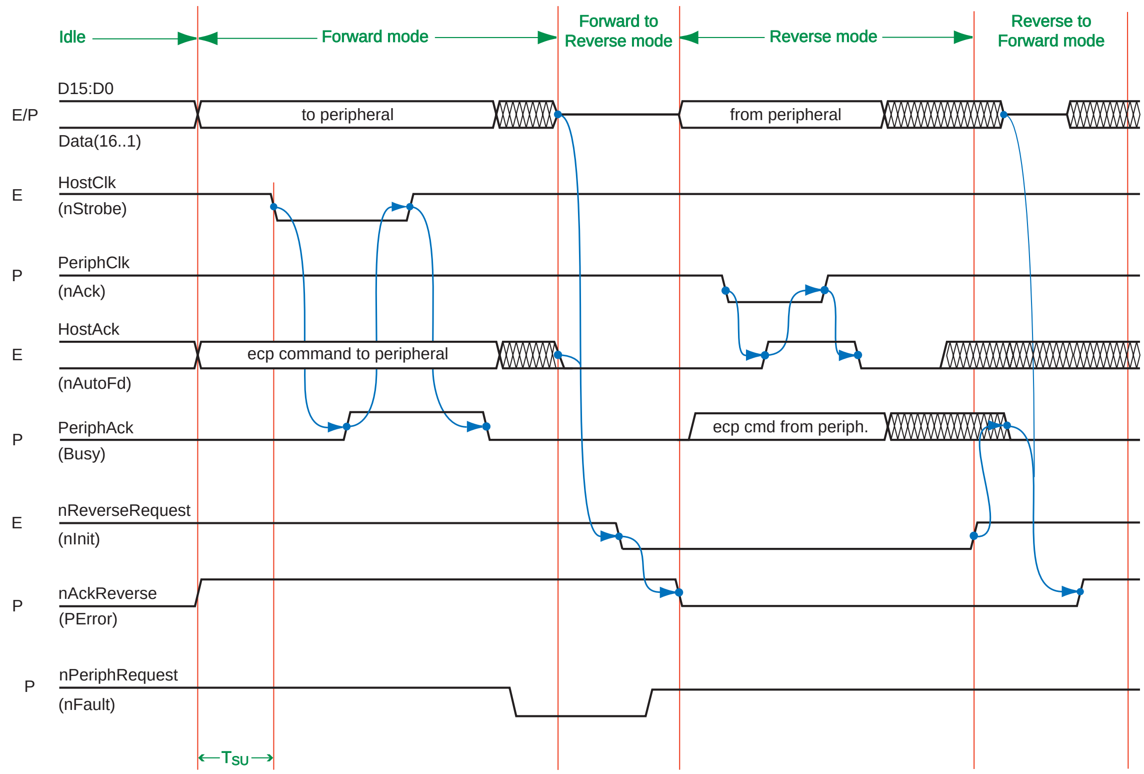
Interface Pin Parallel Port-W	Signal	Description of Signal	Origin of Signal
p0d15 - p0d0	D15:D0	Carries 16-bit data bi-directionally (half-duplex).	Both
p0perror	nAckReverse	Asserted (0) to acknowledge nReverseRequest ($\overline{\text{p0init}}$).	Peripheral
$\overline{\text{p0autofd}}$	HostAck	Provides command/data status in the forward direction. Handshakes with PeriphClk to transfer data in the reverse direction. Set high to indicate a data cycle.	ETRAX 100LX
$\overline{\text{p0strobe}}$	HostClk	Handshakes with PeriphAck to transfer data in the forward direction. Set low to indicate valid data, and set high to clock data to the peripheral.	ETRAX 100LX
$\overline{\text{p0ack}}$	PeriphClk	Handshakes with HostAck to transfer data in the reverse direction.	Peripheral
p0busy	PeriphAck	Handshakes with HostClk to transfer data in the forward direction. Provides command/data status in the reverse direction.	Peripheral
$\overline{\text{p0fault}}$	nPeriphRequest	Set low to indicate that reverse data is available.	Peripheral
$\overline{\text{p0init}}$	nReverseRequest	Set low to switch channel to reverse direction.	ETRAX 100LX

ECP wide (16-bit) register configuration

Only parallel port p0 controls the ECP wide (16-bit) mode. The mode is enabled by bit 10 (wide) in port p0 configuration register R_PAR0_CONFIG and bit 2 (par0), bit 7 (par1) and bit 31 (par_w) in general configuration register R_GEN_CONFIG. The following truth table shows the required bit settings.

R_GEN_CONFIG<2> par0	R_GEN_CONFIG<7> par1	R_GEN_CONFIG<31> par_w	R_PAR0_CONFIG<10> wide	Mode
0	X	0	X	
1	X	0	0	ECP
1	0	1	1	ECP wide

All bit combinations other than those shown above are forbidden.



T_{SU} can be set between 10 ns and 5 µs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host) P = Peripheral

Figure 13-9 ECP Wide (16-bit) Mode Timing

The data lines of both parallel ports are used for data transfers in ECP wide (16-bit) mode. A multiplexer, controlled by the **wide** signal, places the upper byte (D15:D8) of the data word on to the data lines of port p1. A dedicated read/write register R_PAR_ECP_16_DATA stores the last 16-bit data word sent or received.

ECP wide (16-bit) mode time periods

Setup time T_{SU} is configurable in register R_PAR0_DELAY.

13.2.7 EPP Mode

The IEEE Enhanced Parallel Port (EPP) protocol is for high-speed, half-duplex (forward and reverse), parallel data exchange between ETRAX 100LX and a peripheral. All activities are controlled by ETRAX 100LX, which first selects a register within a peripheral, then performs a series of asynchronous read/write operations with the selected register.

The significant signals used by the parallel ports in the EPP mode, and the corresponding signal names at the multiplexed I/O interface, are summarised in the table below.

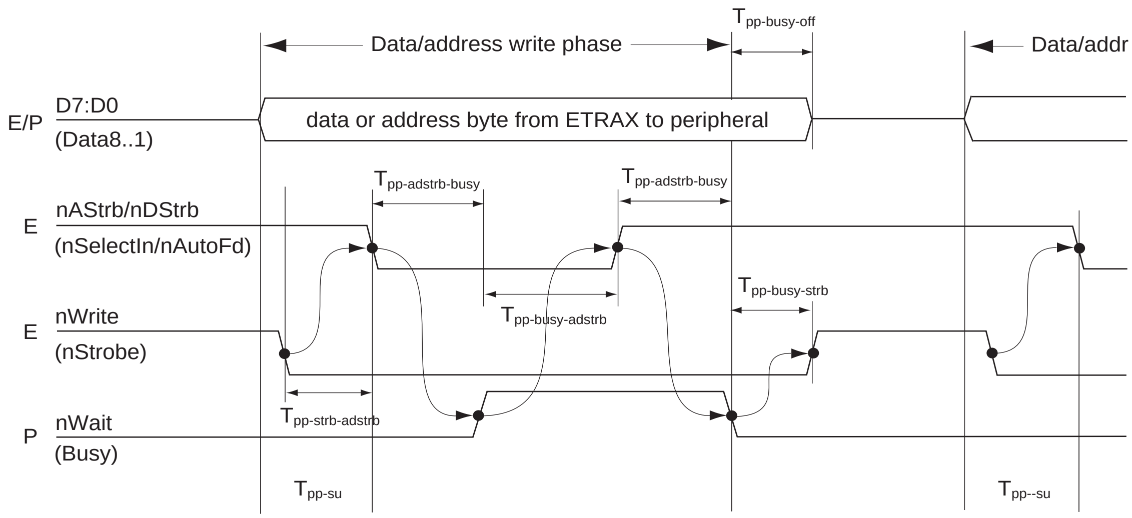
Interface Pin Name		Signal	Description of Signal	Origin of Signal
Port_0	Port_1			
p0d7:p0d0	p1d7:p1d0	D7:D0	Carries 8-bit data/addresses bi-directionally.	Both
$\overline{\text{p0selectin}}$	$\overline{\text{p1selectin}}$	nAStrb	Address strobe. Set low when address bytes are being written or read. Set high when data bytes are being written or read.	ETRAX 100LX
$\overline{\text{p0autofd}}$	$\overline{\text{p1autofd}}$	nDStrb	Data strobe. Set low when data bytes are being written or read. Set high when address bytes are being written or read.	ETRAX 100LX
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	nWrite	Selects write cycles. Set low when address/data bytes are being written. Set high when address/data bytes are being read.	ETRAX 100LX
p0busy	p1busy	nWait	Handshake signal. Set low to indicate that a read or write cycle can commence. Set high to indicate that a read or write cycle can terminate.	Peripheral

EPP write modes

To meet the requirements of the IEEE 1284 EPP protocol, ETRAX 100LX offers three different EPP write modes (1, 2 and 3). The required mode is established in registers R_PAR0_CONFIG and R_PAR1_CONFIG.

In the EPP write modes, ETRAX 100LX asserts the nWrite signal, places a data or address byte on the data bus, and asserts the corresponding strobe signal (nDStrb for data: nAStrb for an address). Within a response time, the peripheral signifies that it is ready to receive the data/address byte by de-asserting the nWait signal. ETRAX 100LX responds by de-asserting nDStrb/nAStrb to latch the data/address into the peripheral device. The peripheral acknowledges the end of the cycle and signifies that it is ready for the next cycle by asserting the nWait signal.

Timing diagrams of EPP write modes 1 to 3 are given in Figures 14-10 to 14-12 that follow.



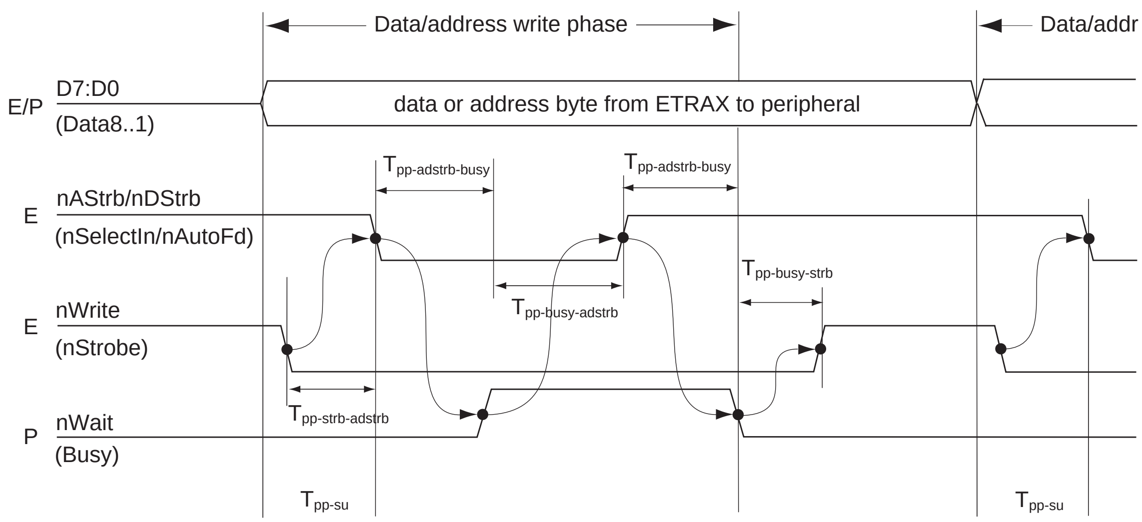
T_{su} can be set between 10 ns and 5 µs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host)   P = Peripheral

Figure 13-10   EPP Address/Data Write Mode 1 Timing



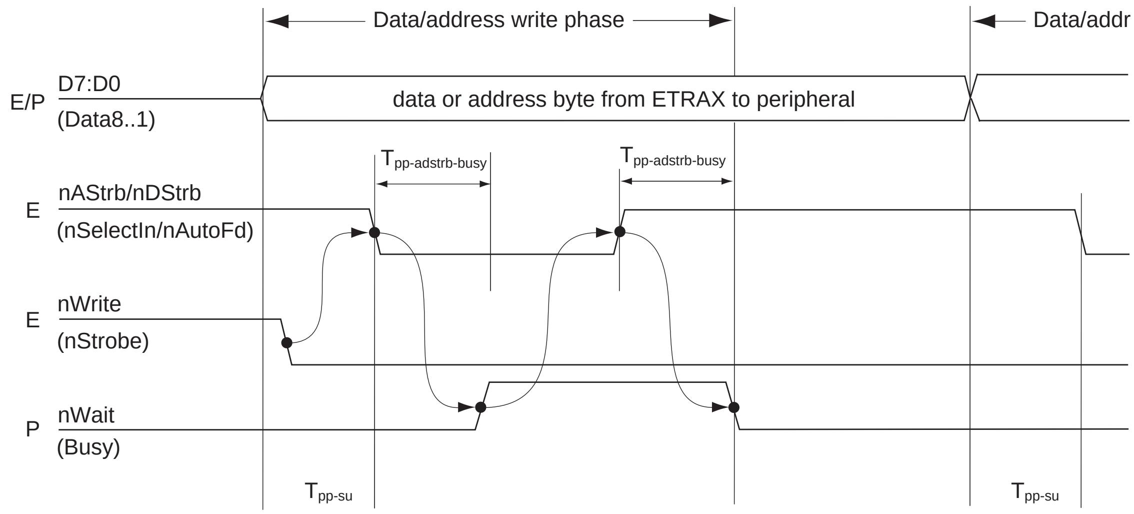
T_{su} can be set between 10 ns and 5 µs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host)   P = Peripheral

Figure 13-11   EPP Address/Data Write Mode 2 Timing



T_{su} can be set between 10 ns and 5 μs in steps of 20 ns.

The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

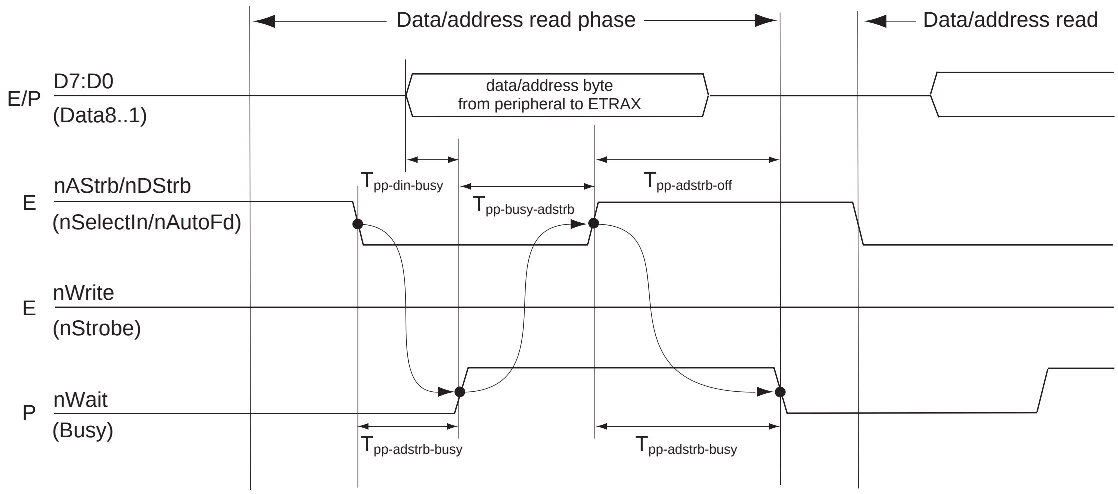
The curved lines represent sequential dependencies.

E = ETRAX 100LX (host)   P = Peripheral

Figure 13-12 EPP Address/Data Write Mode 3 Timing

EPP address/data read cycles

In the EPP address or data write mode, ETRAX 100LX uses the data lines to read an address or data byte from the peripheral. ETRAX 100LX signifies a read cycle by asserting the appropriate strobe signal (nAStrb for an address: nDStrobe for data), and de-asserting the nWrite signal. When the peripheral responds, the strobe (nAStrobe or nDStrobe) is used to latch the address or data byte from the peripheral. The peripheral uses nWait for handshaking at the beginning and end of the write cycle.



The prefix n in a signal name represents a negative (active-low) signal in the IEEE 1284 Standard.

The curved lines represent sequential dependencies.

E = ETRAX 100LX (host) P = Peripheral

Figure 13-13 EPP Address/Data Read Cycle Timing

EPP configuration in the mode registers

The read/write functionality of the EPP mode is implemented by a configuration of the parallel port registers. Bits 11 and 2 to 0 in register R_PAR0_CONFIG or R_PAR1_CONFIG respectively are used to choose the read-write mode, where bits 2 to 0 are the mode field and bit 11 acts as the extended mode selector. The choice of an address or data transfer cycle is realized by bit 3 in the respective configuration register.

13.2.8 Manual Mode

In the context of ETRAX 100LX, Manual mode is the configuration of the parallel ports by means of customised software. This feature enhances the versatility of ETRAX 100LX by permitting the implementation of parallel data transfer protocols not otherwise supported within the IEEE 1284 standard.

To facilitate the Manual mode, all signals to and from the parallel ports are programmable. When Manual mode is operational these discrete signals, and the data buses, are read/written by software through the mode registers.

The software-programmable signals are:

Port p0	Port p1	Port-W
p0perror	p1perror	p0perror
$\overline{\text{p0ack}}$	$\overline{\text{p1ack}}$	$\overline{\text{p0ack}}$
p0busy	p1busy	p0busy
$\overline{\text{p0fault}}$	$\overline{\text{p1fault}}$	$\overline{\text{p0fault}}$
p0select	p1select	p0select
p0data_oe	p0data_oe	p0data_oe
$\overline{\text{p0selectin}}$	$\overline{\text{p1selectin}}$	$\overline{\text{p0selectin}}$
$\overline{\text{p0autofd}}$	$\overline{\text{p1autofd}}$	$\overline{\text{p0autofd}}$
$\overline{\text{p0strobe}}$	$\overline{\text{p1strobe}}$	$\overline{\text{p0strobe}}$
$\overline{\text{p0init}}$	$\overline{\text{p1init}}$	$\overline{\text{p0init}}$
p0d7 - p0d0	p1d7 - p1d0	p0d7 - p0d0
		p0d15 - p0d8

Table 13-1 Software-Controlled Signals in Manual Mode

Please refer to chapter 19.10 *Multiplexed Interfaces* for details of the multiplexed I/O interface that includes these signals.

13.3 Parallel Port Interrupts

Both parallel ports can generate a number of different interrupts, if configured to do so in the Interrupt Mask registers.

13.3.1 Peripheral Interrupt

The parallel ports can be configured so that in the Nibble, Byte, ECP and ECP wide (16-bit) modes, a peripheral can attract the attention of ETRAX 100LX through signal sequences. The peripheral requests attention by means of handshaking between the PtrClk, HostBusy and AckDataReq signals, which generates the interrupt. ETRAX 100LX acknowledges the interrupt when it is ready to receive data.

The peripheral interrupt for parallel port p0 is enabled in the par0_peri field (bit 10) of the R_IRQ_MASK0_SET register. It is cleared by reading the peri_int field (bit 24) in register R_PAR0_CTRL_DATA.

The peripheral interrupt for parallel port p1 is enabled in the par1_peri field (bit 18) of the R_IRQ_MASK1_SET register. It is cleared by reading the peri_int field (bit 24) in register R_PAR1_CTRL_DATA.

13.3.2 ECP Command Interrupt

The parallel ports can be configured to generate an interupt in ECP reverse mode in response to the start of a command cycle from the peripheral.

The ECP command interrupt for parallel port p0 is enabled in the par0_ecp_cmd field (bit 11) of the R_IRQ_MASK0_SET register. It is cleared by reading the ecp_cmd field (bit 8) in register R_PAR0_STATUS_DATA.

The ECP command interrupt for parallel port p1 is enabled in the par1_ecp_cmd field (bit 19) of the R_IRQ_MASK1_SET register. It is cleared by reading the ecp_cmd field (bit 8) in register R_PAR1_STATUS_DATA.

13.3.3 Data Available Interrupt

The parallel ports can be configured to generate an interrupt when a port has input data available. When DMA is used for the data transfer, this interrupt indicates that at least one byte has been received since the interrupt was last cleared.

The data available interrupt for parallel port p0 is enabled in the par0_data field (bit 9) of the R_IRQ_MASK0_SET register. It is cleared by reading the data field (bits 7:0) in register R_PAR0_CTRL_DATA.

The data available interrupt for parallel port p1 is enabled in the par1_data field (bit 17) of the R_IRQ_MASK1_SET register. It is cleared by reading the data field (bits 7:0) in register R_PAR1_CTRL_DATA.

13.3.4 Ready Interrupt

The parallel ports can be configured to generate an interrupt when a port is ready to get new data for transmission.

The ready interrupt for parallel port p0 is enabled in the par0_ready field (bit 8) of the R_IRQ_MASK0_SET register. It is cleared by writing the data field (bits 7:0) in register R_PAR0_CTRL_DATA.

The ready interrupt for parallel port p1 is enabled in the par1_ready field (bit 16) of the R_IRQ_MASK1_SET register. It is cleared by writing the data field (bits 7:0) in register R_PAR1_CTRL_DATA.

Note: 3Note: The par0_ready/par1_ready bits should be masked off when the DMA is used for data transfers.

13.3.5 EPP Interrupts

A peripheral gains the attention of the parallel port by means of a single interrupt request on the port's peripheral acknowledgement line ($\overline{p0ack}$ and $\overline{p1ack}$). To generate an interrupt, the acknowledgement signal is pulsed low for period T_p (0.5 ms). Although an interrupt can be asserted at any time it is independent of, and does not interfere with, the EPP cycles.

The status of the EPP interrupt for parallel port p0 is indicated by the ack field (bit 27) of register R_PAR0_STATUS_DATA and the corresponding field (bit 11) of register R_PAR0_STATUS.

The status of the EPP interrupt for parallel port p1 is indicated by the ack field (bit 27) of register R_PAR1_STATUS_DATA and the corresponding field (bit 11) of register R_PAR1_STATUS.

14 SHARED RAM INTERFACE

The shared RAM interface handles 8-bit and 16-bit transfers between an external device and the system SRAM. The interface operates with standard SRAM. Data transfers are controlled by an asynchronous handshake protocol.

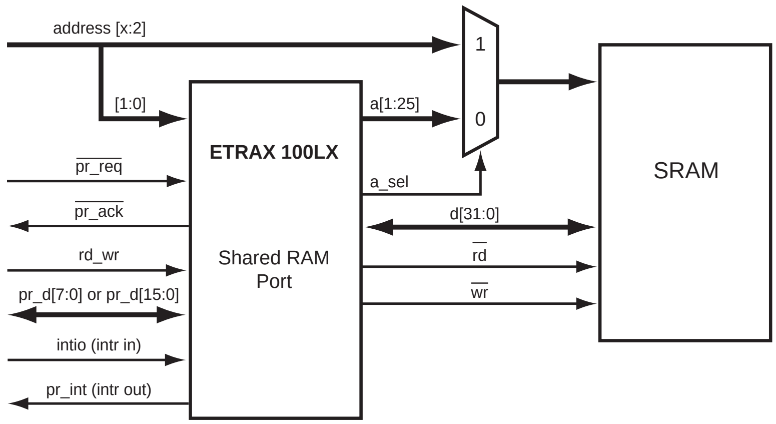


Figure 14-1 How to Connect for Shared RAM

The address from the external device is supplied to the SRAM through external multiplexers, except for address bit 0 and 1 that are multiplexed internally.

Shared RAM interface cycles are like normal bus cycles, except that the a_sel signal on the s0atn pin is high during the cycle. Shared RAM interface cycles are never packed together in bursts. The cycles can be 8-bit or 16-bit wide. 16-bit cycles are always 16-bit aligned.

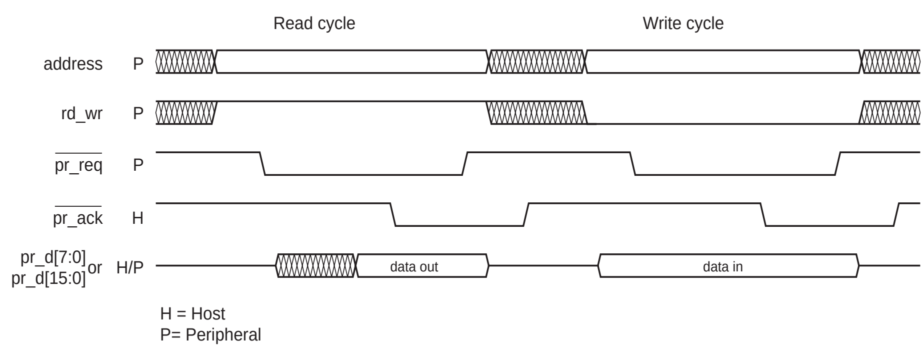


Figure 14-2 Shared RAM Timing

14.1 Shared RAM Interface Configuration

Register	Function
R_SHARED_RAM_CONFIG	A 32-bit write only register to configure and enable the shared RAM interface.
R_SHARED_RAM_ADDR	A 32-bit write only register that sets bits 29-8 of the base address for the shared RAM area.

Table 14-1 Shared RAM Interface Registers

To initiate the Shared RAM interface, first write to the R_GEN_CONFIG register, then to R_SHARED_RAM_ADDR, and finally to R_SHARED_RAM_CONFIG. For more information see chapter 18.5 *Shared RAM Interface Registers*.

14.2 Shared RAM Interrupts

One interrupt signal in each direction is provided. The incoming interrupt set on the Shared RAM interface intio pin is negative edge triggered. The interrupt is cleared by setting the clri bit of the R_SHARED_RAM_CONFIG register.

The interrupt going out, set on the Shared RAM interface pin pr_int, is active low and stays low for 600 ns. This interrupt is generated if the pint field in R_SHARED_RAM_CONFIG is set to int.

15 TIMERS

15.1 General

The timer block consists of:

- A fixed clock divider
- Clock prescaling (i.e. a programmable clock divider)
- Two programmable 8-bit timers: *timer0* and *timer1*
- One watchdog timer

The fixed clock divider is also used as a baud rate generator for the asynchronous and synchronous serial ports. The two programmable timers can be cascaded to form one 16-bit timer.

15.2 Timer Registers

Table 15-1 below provides a brief description of the timer registers. For more detailed information, please see chapter 18.4 *Timer Registers*.

Register	Function
R_TIMER_CTRL	A 32-bit write only register to control the operation, clock selection, and divide factor for timer0 and timer1.
R_TIMER_DATA	A 32-bit read only register for the high and low byte of the fixed clock divider, and the current value of timer0 and timer1.
R_TIMER01_DATA	A 16-bit read only register for the combined count value of timer0 and timer1. Typically used when the timers are cascade coupled.
R_TIMER0_DATA	A byte size read only register for the current count value of timer0.
R_TIMER1_DATA	A byte size read only register for the current count value of timer1.
R_WATCHDOG	A 32-bit write only register for enabling and restarting the watchdog timer.
R_CLOCK_PRESCALE	A 32-bit write only register for the divide factor for timer- and serial clock prescaling.
R_TIMER_PRESCALE	A 16-bit write only register for the divide factor for timer clock prescaling.
R_PRESCALE_STATUS	A 32-bit read only register for the current value of the serial- and timer divide value.
R_TIM_PRESC_STATUS	A 16-bit read only register for the current count value of the timer divide factor.

Table 15-1 Timer registers

15.3 Clock Prescaling: The Programmable Clock Divider

The programmable clock divider divides a 25MHz clock with a 16-bit value. This 16-bit value ranges from 2 to 65535, and results in a new clock available for the timers that ranges from 12.5MHz down to 381.5Hz. The value for the new available clock is written and read in the internal register R_TIMER_PRESCALE.

15.4 Programmable Timers

15.4.1 Timer Operation

The ETRAX 100LX has two programmable timers, timer0 and timer1, which are configured in the register R_TIMER_CTRL. Each one is an 8-bit binary down counter.

Each timer can be loaded with a divide factor between 1 and 256, in R_TIMER_CTRL. When started, the timer:

1. Counts down to 1
2. Generates an interrupt (timer0 or timer1 respectively)
3. Restarts from the programmed divide factor

If the divide factor is changed while the timer is running, it will not take effect until the ongoing count has expired.

Each timer has two mode fields that control its operation:

- timer0: tm0 and i0
- timer1: tm1 and i1

The tm0 or tm1 mode field controls the following:

Timer mode:	Operation:
00	Stop the timer and load it with the divide factor
01	Stop the timer and preserve current count value
10	Run
11	Reserved, do not use

Each timer also has a i0 or i1 (clear interrupt) mode field. Setting it to 1 clears the interrupt, and setting it to 0 has no effect.

The current count value of both timers can be read in R_TIMER_DATA.

15.5 Timer Input Clock

The timer input clock can be individually selected for each timer. In addition to the programmable clock, the following input frequencies can be selected from the fixed clock divider:

Clock sel:	Nominal frequency: (Note 1)
0	0.3 kHz
1	0.6 kHz
2	1.2 kHz
3	2.4 kHz
4	4.8 kHz
5	9.6 kHz
6	19.2 kHz
7	38.4 kHz
8	57.6 kHz
9	115.2 kHz
10	230.4 kHz
11	460.8 kHz
12	921.6 kHz
13	1843.2 kHz
14	6250.0 kHz

Note 1: The actual frequency is 64 ppm higher than the nominal value (given an exact clock reference input), for all frequencies except 6250 kHz.

15.5.1 Timer0 Input Clock

Timer0 can use the external clock by connecting to the general IO pin pb6, or it can use the prescaled clock.

In order to use the prescaled clock (from the programmable clock divider) or the external clock, the clksel0 field of R_TIMER_CTRL must be set to flexible (15), as shown in table 15-2 below.

To use the prescaled clock, presc_ext must be cleared, and to use the external clock presc_ext must be set.

clksel0	presc_ext	clock rate
15	0	timer prescaled clock
15	1	external clock

Table 15-2 clksel0 settings

When using the external clock, timer0 will act as a pulse counter with a maximum of 256 pulses. The external clock must be enabled in R_GEN_CONFIG_II.

Timer0 can also be cascaded with timer1, resulting in a maximum of 65536 pulses.

15.5.2 Timer1 Input Clock

In order to use the prescaled clock (from the programmable clock divider), presc_timer1 of R_TIMER_CTRL must be set:

presc_timer1	clock rate
0	normal (default)
1	timer prescaled clock

Note 2: Setting presc_timer1 in R_TIMER_CTRL overrides all settings on clksel1.

15.6 Cascade Mode

The two programmable counters can be cascaded to form one 16-bit timer. Cascade mode is selected by setting the clksel1 field for timer1 to cascade0. The timer0 interrupt is used to signal the end count of the cascaded counter. The timer1 interrupt is not used in cascade mode, and should be masked off in R_IRQ_MASK0_SET.

When the counters operate in cascade mode, the timer modes, tm0 and tm1, should be set to the same value for both counters with one single write operation.

15.7 Watchdog Timer

When the watchdog timer is started, it generates an NMI if the watchdog is not restarted or stopped within 0.1 s. If it still is not restarted or stopped after an additional 3.3 ms, the watchdog timer resets the chip. The watchdog timer is stopped after reset. The watchdog timer is controlled by the register R_WATCHDOG, which contains an enable field and a 3-bit key value. The effect of writing to the register is described in the table below:

Watchdog	Value written:		
state:	To enable:	To key:	Operation:
stopped	0	X	No effect
stopped	1	key_val	Start watchdog with key = key_val
started	0	~key (Note 3)	Stop watchdog
started	1	~key (Note 3)	Restart watchdog with key = ~key
started	X	new_key_val	Change key to new_key_val

Note 3: '~' is the bitwise NOT operator.

15.8 Timer Interrupts

There are two timer interrupts, one each for timer0 and timer1.

timer1

This interrupt is set whenever timer1 reaches its terminal count. It is cleared by setting the i1 field in R_TIMER_CTRL.

timer0

This interrupt is set whenever timer0 reaches its terminal count. It is cleared by setting the i0 field in timer register R_TIMER_CTRL.

For more information about ETRAX 100LX interrupts, please refer to chapter 17 *Interrupts*.

16 GENERAL I/O PORTS

ETRAX 100LX has two 8-bit ports for handling general input/output signals. They are designated General port PA and General Port PB respectively.

General Port PA is available for general I/O purposes and can also be used to handle externally-generated interrupts. The signals at General Port PB are multiplexed on to the same pins as some other interfaces and therefore these signals are only available when the multiplexed pins are not in use by other interfaces.

In addition to General Ports PA and PB, the multiplexed interfaces make available a number of other discrete pins for general I/O purposes when the pins are not in use by other interfaces. Each of these pins can be considered as a discrete general I/O port.

The general I/O ports are configured in software by means of dedicated registers (See section 16.4 *General I/O Registers*).

Please see chapter 18.7 *General Port Configuration Registers* for detailed information on the general port configuration registers, and chapter 19 *Electrical Information* for information on the multiplexed interfaces.

16.1 General Port PA

Eight signals (pa7 to pa0) are available at General Port PA. The direction and level of each bit is individually configurable in fields dir7 to dir0 (bits 7 to 0 respectively) of register R_PORT_PA_SET. When a direction field is set to 0, the data signal is an input: when a direction field is set to 1 the data signal is an output. The port can be read in register R_PORT_PA_READ.

The pins of General Port PA can supply a maximum current of 12 mA. The port can, therefore, drive light emitting diodes (LED) and similar components.

Please refer to chapter 19.7 *Asynchronous Serial Port 0 Signals*, for more information on the general port configuration registers.

Please refer to chapter 19.6 *General Port PA Signals*, for electrical information on General Port PA.

16.1.1 Interrupts at General Port PA

Each bit in General Port PA can be configured as a masked interrupt as well as a general I/O signal. Eight external interrupts are thus available, all with the same internally generated vector number. The interrupts are level-triggered by active-high signals. Please refer to chapter 17.6 *Masked Interrupts with Internally Generated Vector Numbers* for detailed information on the interrupts at General Port PA.

The interrupts at General Port PA are configured in a set of interrupt mask and status registers. Please refer to chapter 18.13 *Interrupt Mask and Status Registers* for detailed information on the interrupt mask and status registers.

16.2 General Port PB

The configuration of General Port PB is more complex than that of General Port PA because the bits at General Port PB are multi-functional and the pins allocated to the port are multiplexed with signals from the following interfaces:

- I2C Port
- Peripheral chip-select (CSP) Port
- SCSI Ports
- Synchronous Serial Ports
- USB Ports

General Port pb6 can also be used as an external baud rate clock input. In this case, the port should be configured as a general port input. For more information see chapter 11 *Asynchronous Serial Ports*.

Only one function may be used on a pin at one time, and the configuration of these pins for General Port PB is prioritized. The general port has the lowest priority, then the CSP Port, then the other interfaces that use the pins as shown in the table below:

Lowest Priority	Next Highest Priority	Next Highest Priority	Highest Priority
General Port PB pb0			I2C Port i2c_d
General Port PB pb1			I2C Port i2c_clk
General Port PB pb2	CSP $\overline{\text{csp1}}$		USB Port p1 usb1_vpo
General Port PB pb3	CSP $\overline{\text{csp2}}$		USB Port p1 usb1_vmo
General Port PB pb4	CSP $\overline{\text{csp3}}$	Synch. Serial Port p1 ss1_out3 ss1_in3	SCSI-8 port p0 SCSI-W Port $\overline{\text{s0enph}}$
General Port PB pb5	CSP $\overline{\text{csp5}}$		USB Port p1 usb1_vm
General Port PB External baud rate clock input pb6	CSP $\overline{\text{csp6}}$		
General Port PB pb7	CSP $\overline{\text{csp7}}$	Synch. Serial Port p3 ss3_out3 ss3_in3	SCSI-8 Port p1 SCSI-W Port s0en1oid $\overline{\text{s1enph}}$

Please refer to chapter 19.10 *Multiplexed Interfaces* for details of the multiplexed interfaces.

16.2.1 Configuration of Signal Directions at General Port PB

When General Port PB is in use, each bit is individually configurable in register R_PORT_PB_SET and can always be read in register R_PORT_PB_READ. Fields dir7 to dir0 (bits 15 to 8 respectively) in R_PORT_PB_SET establish the direction of the corresponding data signal at General Port PB. When a direction field is set to 0, the data signal is an input: when a direction field is set to 1 the data signal is an output.

16.2.2 General Port PB and the I2C Interface

Bits pb0 and pb1 of General Port PB can be used by the I2C interface, making it possible to run the I2C interface without conflict with other bits of the General Port PB byte. The I2C interface can use pin V15 for data signal i2c_d and pin W16 for clock signal i2c_clk. Consequently, if the I2C interface is configured and enabled, bits pb0 and pb1 of General Port PB are unavailable for I/O purposes.

Selection of the I2C mode is made by the i2c_en (bit 27) and $\overline{\text{i2c\_oe}}$ (bit 24) fields of register R_PORT_PB_SET in accordance with the following table.

i2c_en	**$\overline{\text{i2c\_oe}}$**	**Pin V15**	**Direction**	**Pin W16**	**Direction**
0	X	pb0	in/out	pb1	in/out
1	0	i2c_d	out	i2c_clk	out
1	1	i2c_d	in	i2c_clk	out

16.2.3 General Port PB and the Peripheral Chip-Select Signals

The peripheral chip-select (CSP) Port can use any or all of the upper six bits (pb7 to pb2) of General Port PB. Each of these bits can be set separately by chip-select enable fields (cs7 to cs2) in register R_PORT_PB_SET. If a chip-select enable field is set to 0, then the corresponding bit is configured as a general I/O signal. If a chip-select enable field is set to 1, then the corresponding bit is configured as a chip-select signal and is unavailable for general I/O purposes.

When chip-select signals are in use, General I/O pins pb7 to pb5 become $\overline{\text{csp7}}$ to $\overline{\text{csp5}}$ and pins pb4 to pb2 become $\overline{\text{csp3}}$ to $\overline{\text{csp1}}$. After system reset, these pins are inputs so they require the connection of external pull-up resistors. Note that pins $\overline{\text{csp0}}$ and $\overline{\text{csp4}}$ are always available, and are always outputs.

The effects of certain chip-select enable fields are overridden if the following are enabled:

- Synchronous Serial Ports (see 16.2.4);
- SCSI Ports (see 16.2.5);
- USB Ports (see 16.2.6).

16.2.4 General Port PB and the Synchronous Serial Ports

In some of their modes, Synchronous Serial Ports p1 and p3 use General Port PB bits pb4 and pb7 respectively. The syncser1 and syncser3 fields of the R_PORT_PB_SET register configure the connection of pins pb4 and pb7 to the synchronous serial ports.

The effect of chip-select enable field cs7 is overridden by the syncser3 field in R_PORT_PB_SET. Similarly field cs4 is overridden by the syncser1 field in R_PORT_PB_SET.

A configuration of pb7 for use with SCSI-8 Port p1 or SCSI-W will override the syncser3 field in R_PORT_PB_SET. Similarly, a configuration of pb4 for use with SCSI-8 port p0 or SCSI-W will override the syncser1 field in R_PORT_PB_SET.

16.2.5 General Port PB and the SCSI Ports

The SCSI-8 Ports p0 and p1 use General Port PB bits pb4 and pb7 respectively. The SCSI-W Port uses both pb4 and pb7. The scsi0 and scsi1 fields of the R_PORT_PB_SET register configure the connection of pins pb4 and pb7 to the SCSI ports.

The effect of chip-select enable field cs7 is overridden by the scsi1 field in R_PORT_PB_SET. Similarly field cs4 is overridden by the scsi0 field in R_PORT_PB_SET.

16.2.6 General Port PB and the USB Ports

USB Port p1 uses three of the pins allocated to General Port PB. Consequently, when the USB port is enabled, bits pb2, pb3 and pb5 of General Port PB are unavailable for general I/O purposes.

Similarly, when USB Port p1 is active, the effect of chip-select enable fields cs1, cs2 and cs5 are automatically overridden.

16.3 Discrete General Ports

Some of the pins that are not used in a chosen configuration of ETRAX 100LX are available as discrete I/O ports. The signals available at these ports are designated gn, where n is a signal number within the range 0 to 31.

When a discrete I/O port is available, its signal direction and/or data are individually configurable. For some of the discrete I/O ports, separate pins are used for the input and output signals. These ports are:

- g1 to g7
- g25 to g31

The signal directions at the other discrete I/O ports can be set in register R_GEN_CONFIG. Fields g24dir (bit 27), g16_23dir (bit 26), g8_15dir (bit 25) and g0dir (bit 24) establish the signal direction at the corresponding discrete I/O port. When a direction field is set to 0, the signal is an input: when a direction field is set to 1 the signal is an output.

The discrete I/O ports are written and read in the data field (bits 31 to 0) of register R_PORT_G_DATA.

For detailed information on these miscellaneous I/O ports, please refer to chapter 19.9 *Multiplexed Signal Groups*, which provides pin-by-pin listings for the multiplexed interfaces.

16.4 General I/O Registers

As noted above, the general I/O ports are served by a set of dedicated configuration registers. The table below summarizes the purpose of each register.

Register	Function
R_PORT_PA_SET comprising:	A 32-bit, write-only register in which the upper 16 bits are reserved.
R_PORT_PA_DATA	An 8-bit, write-only register formed from the lowest byte of the lower 16 bits of PA_PORT_PA_SET. The register contains the data_out byte to General Port PA.
R_PORT_PA_DIR	An 8-bit, write-only register formed from the high byte of the lower 16 bits of PA_PORT_PA_SET. Each bit sets the direction (in or out) of a corresponding signal to or from General Port PA.
R_PORT_PA_READ	A 32-bit, read-only register in which the upper 24 bits are reserved. The 8 low bits contain the data_in byte from General Port PA.
R_PORT_PB_SET comprising:	A 32-bit, write-only register in which the upper two bits are reserved.
R_PORT_PB_DATA	An 8-bit, write-only register formed from the lowest byte of PA_PORT_PB_SET. The register contains the data_out byte to General Port PB.
R_PORT_PB_DIR	An 8-bit, write-only register formed from the second lowest byte of PA_PORT_PB_SET. Each bit sets the direction (in or out) of a corresponding signal to or from General Port PB.
R_PORT_PB_CONFIG	An 8-bit, write-only register formed from the third lowest byte of PA_PORT_PB_SET. Six bits are used to configure pb7 to pb2 as chip-select or general I/O signals, and the other two bits determine whether bits pb7 and pb4 are used to carry SCSI signals.
R_PORT_PB_I2C	An 8-bit, write-only register formed from the highest byte of PA_PORT_PB_SET, but with the upper two bits reserved. Four bits are used to configure the I2C interface, and the other two bits determine whether bits pb7 and pb4 are used to carry synchronous serial port signals.
R_PORT_PB_READ	A 32-bit, read-only register in which the upper 24 bits are reserved. The 8 lowest bits contain the data_in byte from General Port PB.

For detailed information on the general port configuration registers, please refer to chapter 18.6 *General Configuration Registers*.

17 INTERRUPTS

Most of the interfaces of the ETRAX 100LX generate interrupts. Three types of interrupts are supported:

- Non-maskable interrupts (NMI);
- Maskable interrupts with internally generated vector numbers;
- Maskable interrupts with external vector numbers.

The NMI have the highest priority, and the maskable interrupts with external vector numbers have the lowest priority. All maskable interrupts with internally generated vector numbers have the same priority.

17.1 Interrupt Masks

There are two levels of interrupt masks:

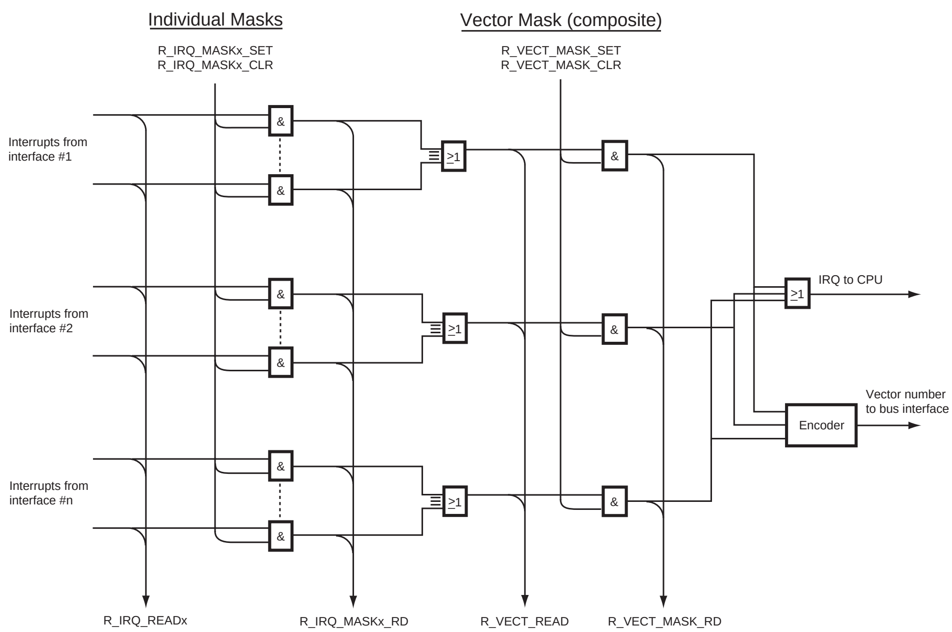


Figure 17-1 Interrupt Masks

At one level, the individual bits of all interrupts except the NMI are masked. At the other level there is one composite mask bit for each vector number (i.e. one composite bit for each different internal interface). There are separate addresses for setting and clearing the mask bits and only asserted mask bits (those set to 1), are set or cleared (see section *17.4 Interrupt Registers*).

It is recommended that the software driver for each interface should control its own mask bits at the individual interrupt masking level, while general system functions control the composite vector number mask. Thus there is one mask bit on each level, even for those vectors that have one interrupt only.

All interrupt masks are undefined after reset. They must be initialised before the interrupt is enabled in the CPU.

17.2 Interrupt Status

The status of the individual interrupts can be read before and after they are gated with the interrupt mask. Registers R_IRQ_READ0 to R_IRQ_READ2 and R_USB_IRQ_RD contain the interrupt bits prior to individual masking. Registers R_IRQ_MASK0_RD to R_IRQ_MASK2_RD and R_USB_IRQ_MASK_RD contain the interrupt bits after the individual mask.

The status of the interrupt vector numbers can be read before and after they are gated with the composite mask. Register R_VECT_READ shows the interrupt vector number bits prior to composite masking. Register R_VECT_MASK_RD shows the interrupt vector number bits after the composite mask.

17.3 USB Interrupts

The USB is capable of generating a relatively large number of maskable interrupts with internally generated vector numbers. Consequently the USB interrupts are handled within the set of registers dedicated to the USB, as distinct from the set of registers entirely dedicated to interrupt handling.

17.4 Interrupt Registers

The interrupt capability of ETRAX 100LX is supported by a dedicated set of interrupt mask and status registers. These registers are arranged in three sub-sets (designated 0 to 2) that contain different interrupt bits at the individual masking level, and a fourth sub-set for the vector number bits at the composite masking level. As previously noted, a sub-set of the R_USB register set contains the USB interrupt bits at the individual masking level. This architecture is summarized in the tables below.

Register Sub-Set 0	Register	Purpose
Handles the NMI and maskable interrupts for: Ethernet error and statistics counters Network ATA Parallel port p0 Parallel port W Shared RAM SCSI-8 port p0 SCSI port W External interrupt External DMA Timers	R_IRQ_MASK0_SET	A 32-bit, write-only register for setting the 29 maskable interrupt fields with internally generated vector numbers. Four fields handle multiple interrupt mask bits by means of the multiplexed interfaces.
	R_IRQ_READ0	A 32-bit, read-only register that contains the NMI and maskable interrupts. It shows the status of these interrupts prior to the individual mask.
	R_IRQ_MASK0_RD	A 32-bit, read-only register that contains the NMI and maskable interrupts. It shows the status of these interrupts after the individual mask.
	R_IRQ_MASK0_CLR	A 32-bit, write-only register for clearing the mask bits of the maskable interrupts that have been set in register R_IRQ_MASK0_SET.

Table 17-1 Interrupt registers: Register sub-set 0

Register Sub-Set 1	Register	Purpose
Handles SW-generated interrupts and the maskable interrupts for: Parallel port p1 SCSI-8 port p1 Async. serial ports Sync. serial ports General port PA	R_IRQ_MASK1_SET	A 32-bit, write-only register for setting the maskable interrupts with internally generated vector numbers. Of these, one field handles two interrupt mask bits by means of the multiplexed interfaces.
	R_IRQ_READ1	A 32-bit, read-only register that contains the maskable interrupts. It shows the status of these interrupts prior to the individual mask.
	R_IRQ_MASK1_RD	A 32-bit, read-only register that contains the maskable interrupts. It shows the status of these interrupts after the individual mask.
	R_IRQ_MASK1_CLR	A 32-bit, write-only register for clearing the mask bits of the maskable interrupts that have been set in register R_IRQ_MASK1_SET.

Table 17-2 Interrupt registers: Register sub-set 1

Register Sub-Set 2	Register	Purpose
Handles all the interrupts for the DMA channels.	R_IRQ_MASK2_SET	A 32-bit, write-only register for setting the maskable DMA channel interrupts with internally generated vector numbers.
	R_IRQ_READ2	A 32-bit, read-only register for reading the maskable DMA interrupts. It shows the status of these interrupts prior to the individual mask.
	R_IRQ_MASK2_RD	A 32-bit, read-only register that contains the maskable interrupts. It shows the status of these interrupts after the individual mask.
	R_IRQ_MASK2_CLR	A 32-bit, write-only register for clearing the mask bits of the maskable interrupts that have been set in register R_IRQ_MASK2_SET.

Table 17-3 Interrupt registers: Register sub-set 2

USB Register Sub-Set	Register	Purpose
Handles all the interrupts for the USB.	R_USB_IRQ_MASK_SET	A 16-bit, write-only register for setting the maskable USB interrupts with internally generated vector numbers.
	R_USB_IRQ_RD	A 16-bit, read-only register for reading the maskable interrupts. It shows the status of these interrupts prior to the individual mask.
	R_USB_IRQ_MASK_RD	A 16-bit, read-only register that contains the maskable interrupts. It shows the status of these interrupts after the individual mask.
	R_USB_IRQ_MASK_CLR	A 32-bit, write-only register for clearing the mask bits of the maskable interrupts that have been set in register R_USB_IRQ_MASK_SET.

Table 17-4 Interrupt registers: USB Register sub-set

Vector Register Sub-Set	Register	Purpose
Handles the interrupt vector number bits at the composite masking level.	R_VECT_MASK_SET	A 32-bit, write-only register for setting the interrupt vector number bits.
	R_VECT_READ	A 32-bit, read-only register for reading the interrupt vector number bits. It shows the status of these bits prior to the composite mask.
	R_VECT_MASK_RD	A 32-bit, read-only register for reading the interrupt vector number bits. It shows the status of these bits after the composite mask.
	R_VECT_MASK_CLR	A 32-bit, write-only register for clearing the interrupt vector number bits that have been set in register R_VECT_MASK_SET.

Table 17-5 Interrupt registers: Vector register sub-set

Please see chapter 18.13 *Interrupt Mask and Status Registers* for detailed information on the interrupt mask and status registers, and 18.16 *Universal Serial Bus Interface Control Registers* for USB registers.

17.5 Non-Maskable Interrupts

There are two sources of non-maskable interrupt:

- The $\overline{\text{nmi}}$ pin:
- The watchdog timer.

The source of an NMI request can be identified by reading registers R_IRQ_MASK0_RD or R_IRQ_READ0.

Both NMI have the internally generated vector number 0x21.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
nmi_pin	31	This interrupt is from the external $\overline{\text{nmi}}$ pin.	This interrupt is cleared in the external unit.
watchdog_nmi	30	This interrupt is from watchdog timer.	This interrupt is cleared by stopping or restarting the watchdog timer.

Table 17-6 Non-maskable interrupts

17.6 Masked Interrupts with Internally Generated Vector Numbers

The internal interfaces of ETRAX 100LX that are not mutually exclusive have individual, internally generated vector numbers. Mutually exclusive interfaces (e.g. SCSI-8 p0 and parallel port p0), may have the same vector number. Each interface has one or more different interrupts, all of which share the same vector number. If two or more interfaces issue an interrupt at the same time, vector number 0x2F is generated and the different inputs are sorted by the software. See table 17-7 below:

Vector number	Number of interrupts	Name	Explanation
0x20	-	-	Hardware breakpoint/single step bus fault. For more information, refer to chapter 2.9 *Hardware Breakpoint Mechanism*.
0x21	2	nmi	Non-maskable interrupts (see table 17-6).
0x22	1	timer0	Timer 0 interrupt (see table 17-13)
0x23	1	timer1	Timer 1 interrupt (see table 17-13)
0x24	9	scsi0_par0	Multiple/multiplexed interrupts for ATA, parallel port p0, shared RAM and SCSI-8 p0 (see table 17-10).
0x25	4	scsi1_par1	Multiple/multiplexed interrupts for parallel port p1 and SCSI-8 p1 (see table 17-15).
0x26	4	network	Network interrupts (see table 17-9).
0x27	10	snmp	Ethernet error and statistics counter interrupts (see table 17-8).
0x28	8	serial	Asynchronous and synchronous serial port interrupts (see table 17-16).
0x29	8	sw	Software generated interrupts (see table 17-14).
0x2A	1	irq_intnr	$\overline{\text{irq}}$ pin when used with internally generated vector number (see table 17-11).
0x2B	8	pa	General port PA interrupts (see table 17-17).
0x2C	1	ext_dma0	External DMA channel 0 interrupt (see table 17-12).
0x2D	1	ext_dma1	External DMA channel 1 interrupt (see table 17-12).
0x2E	-	-	MMU bus fault.
0x2F	-	-	Simultaneous interrupts for more than one vector are active.
0x30	2	dma0	DMA channel 0 interrupts (see table 17-19).
0x31	2	dma1	DMA channel 1 interrupts (see table 17-19).
0x32	2	dma2	DMA channel 2 interrupts (see table 17-19).
0x33	2	dma3	DMA channel 3 interrupts (see table 17-19).
0x34	2	dma4	DMA channel 4 interrupts (see table 17-19).
0x35	2	dma5	DMA channel 5 interrupts (see table 17-19).
0x36	2	dma6	DMA channel 6 interrupts (see table 17-19).
0x37	2	dma7	DMA channel 7 interrupts (see table 17-19).
0x38	6	dma8	Interrupts for DMA channel 8 (see table 17-19), and for sub-channels 8.0 to 8.3 (see table 17-18).
0x39	2	dma9	DMA channel 9 interrupts (see table 17-19).
0x3F	10	usb	USB interface interrupts (see table 17-20).

Table 17-7 Masked Interrupts with Internally Generated Vector Numbers

If a source interrupt is cleared while the CPU is executing its interrupt acknowledgement sequence, the vector number of the interrupting interface is nevertheless generated. However the interrupts are not present in the interrupt status registers.

17.6.1 Interrupts in Register Sub-Set 0

In register sub-set 0, the interrupt mask bits are set in register R_IRQ_MASK0_SET, where only the asserted bits are set (bits written with 0 are not affected). The masked interrupts are read in registers R_IRQ_MASK0_RD. Register R_IRQ_READ0 contains the interrupt bits before they are individually masked. The asserted mask bits are cleared in register R_IRQ_MASK0_CLR.

Each interrupt field occupies the same bit number in all the registers of sub-set 0.

Ethernet Error and Statistics Counter Interrupts

Register sub-set 0 handles ten interrupts associated with Ethernet error and statistics counters. They all have the internally generated vector number 0x27.

Name of interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
sqe_test_error	29	This interrupt is set when the sqe_test_error counter attains the value 128.	This interrupt is cleared by reading the sqe_test_error field of network interface register R_PHY_COUNTERS.
carrier_loss	28	This interrupt is set when the carrier_loss counter attains the value 128.	This interrupt is cleared by reading the carrier_loss field of network interface register R_PHY_COUNTERS.
deferred	27	This interrupt is set when the deferred counter attains the value 128.	This interrupt is cleared by reading the deferred field of network interface register R_TR_COUNTERS.
late_col	26	This interrupt is set when the late_col counter attains the value 128.	This interrupt is cleared by reading the late_col field of the R_TR_COUNTERS register.
multiple_col	25	This interrupt is set when the multiple_col counter attains the value 128.	This interrupt is cleared by reading the multiple_col field of network interface register R_TR_COUNTERS.
single_col	24	This interrupt is set when the single_col counter attains the value 128.	This interrupt is cleared by reading the single_col field of network interface register R_TR_COUNTERS.
congestion	23	This interrupt is set when the congestion counter attains the value 128.	This interrupt is cleared by reading the congestion field of network interface register R_REC_COUNTERS, an action which also clears the overrun interrupt.
oversize	22	This interrupt is set when the oversize counter attains the value 128.	This interrupt is cleared by reading the oversize field of network interface register R_REC_COUNTERS.
alignment_error	21	This interrupt is set when the alignment_error counter attains the value 128.	This interrupt is cleared by reading the alignment_error field of network interface register R_REC_COUNTERS.
crc_error	20	This interrupt is set when the crc_error counter attains the value 128.	This interrupt is cleared by reading the crc_error field of network interface register R_REC_COUNTERS.

Table 17-8 Ethernet Error and Statistics Counter Interrupts

Network Interrupts

Register sub-set 0 contains four network interrupts, all with the internally generated vector number 0x26.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
overrun	19	This interrupt is set when the network receiver experiences a FIFO overrun condition (congestion error). Two interrupts are available (overrun and congestion), but usually only one of them should be enabled. The overrun interrupt should be used if software intervention is necessary when an overrun error occurs. The congestion interrupt should be used if the only action needed is an error count.	This interrupt is cleared by reading the congestion field of network interface register R_REC_COUNTERS, an action which also clears the congestion interrupt.
underrun	18	This interrupt is set when the network transmitter experiences a FIFO underrun condition.	This interrupt is cleared by setting the clr_error field in network interface register R_NETWORK_TR_CTRL.
excessive_col	17	This interrupt is set when the network transmitter experiences collisions for 16 consecutive transmission attempts. It is set after the first collision if the retry field in network interface register R_NETWORK_TR_CTRL is set to disable, or when the transmitter stops after the cancel field of R_NETWORK_TR_CTRL has been set.	This interrupt is cleared by setting the clr_error field in network interface register R_NETWORK_TR_CTRL.
mdio	16	This interrupt is from the MII mdio pin. It is generated when the mdio pin is low. The interrupt should be masked off during normal data transfers over the mdio interface.	This interrupt should be cleared in the external unit that is driving the mdio pin.

Table 17-9 Network Interrupts

Masked Interrupts for ATA, Parallel Port p0, Shared RAM and SCSI-8 Port p0

In register sub-set 0 there are nine different interrupt fields that have the internally generated vector number 0x24. Four of these fields are multi-functional (bits 11 to 8), containing different interface-dependent interrupts.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
ata_drq3	15	This interrupt is set when a unit on ATA bus 3 requests a DMA transfer.	This interrupt is cleared at the end of the DMA transfer on ATA bus 3.
ata_drq2	14	This interrupt is set when a unit on ATA bus 2 requests a DMA transfer.	This interrupt is cleared at the end of the DMA transfer on ATA bus 2.
ata_drq1	13	This interrupt is set when a unit on ATA bus 1 requests a DMA transfer.	This interrupt is cleared at the end of the DMA transfer on ATA bus 1.
ata_drq0	12	This interrupt is set when a unit on ATA bus 0 requests a DMA transfer.	This interrupt is cleared at the end of the DMA transfer on ATA bus 0.
par0_ecp_cmd ata_irq3	11	When parallel port p0 is in ECP mode, this interrupt is set when an ECP command is received at the port. When ATA is in use, this interrupt is set when a unit on ATA bus 3 requests an interrupt.	This interrupt is cleared by reading the ecp_cmd_bit field in parallel port p0 register R_PAR0_DATA. This interrupt is cleared in the external unit on ATA bus 3.
par0_peri ata_irq2	10	When parallel port p0 is in use, this interrupt is set by the peripheral connected to the port. When ATA is in use, this interrupt is set when a unit on ATA bus 2 requests an interrupt.	This interrupt is cleared by acknowledging the peri_int field in parallel port p0 register R_PAR0_CTRL_DATA. This interrupt is cleared in the external unit on ATA bus 2.
par0_data ata_irq1	9	When parallel port p0 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared. When ATA is in use, this interrupt is set when a unit on ATA bus 1 requests an interrupt.	This interrupt is cleared by reading the data field of parallel port p0 register R_PAR0_DATA. This interrupt is cleared in the external unit on ATA bus 1.
par0_ready ata_irq0 mio scsi0	8	When parallel port par0 is in use, this interrupt is set when the port is ready to get new data for transmission. When ATA is in use, this interrupt is set when a unit on ATA bus 0 requests an interrupt. This interrupt is set on the $\overline{\text{intio}}$ pin of the shared RAM interface. When SCSI-8 port p0 is in use, this interrupt is set when the SCSI controller requests service from the CPU.	The interrupt is cleared by writing new data to the data field of parallel port register R_PAR0_DATA. This field should be masked when the DMA is used for data transfers. The interrupt is cleared in the external unit on ATA bus 0. This interrupt is cleared by setting the i field of register R_SHARED_RAM_CONFIG. The interrupt is cleared by writing to the clr_status field (bit 24) of register R_SCSI0_CMD_DATA.
ata_dmaend	7	This interrupt is set when the selected ATA unit releases its DMA request (transfer completed). This interrupt should be masked, except when an ATA DMA transfer is started.	The interrupt is cleared when next ATA DMA transfer commences.

Table 17-10 Masked Interrupts for ATA, Parallel Port p0, Shared RAM and SCSI-8 Port p0

External Interrupt Configuration Fields

In register sub-set 0 there are two fields for configuring the external interrupt on the irq pin, as shown in the table below.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
irq_ext_vector_nr	5	This field configures the external interrupt on the irq pin to have an external vector number. The mask bit is only effective if the irq_int_vector_nr mask bit (4) is cleared.	This interrupt is cleared in the external unit. This can be done by software or the inta output.
irq_int_vector_nr	4	This field configures the external interrupt on the irq pin to have the internally generated vector number 0x2A. When the corresponding mask bit is set, the internally generated interrupt vector number is used and the irq_ext_vector_nr mask bit (5) has no effect.	This interrupt is cleared in the external unit. This can be done by software only.

Table 17-11 External Interrupt Configuration Fields

External DMA Interrupts

In register sub-set 0 there are two fields for handling external DMA interrupts.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
ext_dma1	3	This interrupt has the internally generated vector number 0x2D. It is set when external DMA channel 1 has stopped.	This interrupt should be masked, except when waiting for the completion of a transfer on external DMA channel 1.
ext_dma0	2	This interrupt has the internally generated vector number 0x2C. It is set when external DMA channel 0 has stopped.	This interrupt should be masked, except when waiting for the completion of a transfer on external DMA channel 0.

Table 17-12 External DMA Interrupts

Timer Interrupts

In register sub-set 0 there are two fields for handling the timer interrupts.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
timer1	1	This interrupt has the internally generated vector number 0x23. It is set whenever timer1 reaches its terminal count.	This interrupt is cleared by setting the i1 bit in timer register R_TIMER_CTRL.
timer0	0	This interrupt has the internally generated vector number 0x22. It is set whenever timer0 reaches its terminal count.	This interrupt is cleared by setting the i0 bit in timer register R_TIMER_CTRL.

Table 17-13 Timer Interrupts

17.6.2 Interrupts in Register Sub-Set 1

In this sub-set the masked interrupts are set in register R_IRQ_MASK1_SET and read in register R_IRQ_MASK1_RD. Register R_IRQ_READ1 contains the interrupt bits before they are individually masked. The asserted mask bits are cleared in register R_IRQ_MASK1_CLR.

Each interrupt field occupies the same bit number in all the registers of sub-set 1.

Software Generated Interrupts

Register sub-set 1 has eight fields for handling software generated interrupts. They all have the internally generated vector number 0x29.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
sw_int7	31	This interrupt is generated in software when mask bit sw_int7 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int7 field in register R_IRQ_MASK1_CLR is set.
sw_int6	30	This interrupt is generated in software when mask bit sw_int6 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int6 field in register R_IRQ_MASK1_CLR is set.
sw_int5	29	This interrupt is generated in software when mask bit sw_int5 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int5 field in register R_IRQ_MASK1_CLR is set.
sw_int4	28	This interrupt is generated in software when mask bit sw_int4 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int4 field in register R_IRQ_MASK1_CLR is set.
sw_int3	27	This interrupt is generated in software when mask bit sw_int3 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int3 field in register R_IRQ_MASK1_CLR is set.
sw_int2	26	This interrupt is generated in software when mask bit sw_int2 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int2 field in register R_IRQ_MASK1_CLR is set.
sw_int1	25	This interrupt is generated in software when mask bit sw_int1 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int1 field in register R_IRQ_MASK1_CLR is set.
sw_int0	24	This interrupt is generated in software when mask bit sw_int0 in register R_IRQ_MASK1_SET is asserted.	This interrupt is cleared when the sw_int0 field in register R_IRQ_MASK1_CLR is set.

Table 17-14 Software Generated Interrupts

Please refer to section 17.8 *Software Interrupts* below for more information on the software interrupts.

Parallel Port p1 and SCSI-8 Port p1 Interrupts

Register sub-set 1 has four fields for handling the interrupts at parallel port p1. One field (bit 16), contains an interrupt bit for SCSI-8 port p1 also. All these interrupts have the internally generated vector number 0x25.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
par1_ecp_cmd	19	When parallel port p1 is in ECP mode, this interrupt is set when an ECP command is received at the port.	This interrupt is cleared by reading the ecp_cmd_bit field in parallel port p1 register R_PAR1_CTRL_DATA.
par1_peri	18	When parallel port p1 is in use, this interrupt is set by the peripheral connected to the port.	This interrupt is cleared by acknowledging the peri_int field in parallel port p1 register R_PAR1_CTRL_DATA.
par1_data	17	When parallel port p1 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.	This interrupt is cleared by reading the data field of parallel port p1 register R_PAR1_CTRL_DATA.
par1_ready scsi1	16	When parallel port p1 is in use, this interrupt is set when the port is ready to get new data for transmission. When SCSI-8 port p1 is in use, this interrupt is set when the SCSI controller requests service from the CPU.	The interrupt is cleared by writing new data to the data field of parallel port p1 register R_PAR1_CTRL_DATA. This field should be masked when the DMA is used for data transfers. The interrupt is cleared by writing to the clr_status field (bit 24) of register R_SCSI1_CMD_DATA.

Table 17-15 Parallel Port p1 and SCSI-8 Port p1 Interrupts

Asynchronous and Synchronous Serial Port Interrupts

Register sub-set 1 has eight fields for handling the interrupts of the four asynchronous serial ports (p3 to p0). All of these interrupts have the internally generated vector number 0x28.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
ser3_ready	15	This interrupt is set when Async/Sync serial port p3 is ready to acquire new data for transmission. It should be masked when the DMA is used for data transfers.	This interrupt is cleared by writing new data to the R_SERIAL3_TR_DATA register, or to the data_out field of R_SERIAL3_CTRL.
ser3_data	14	When Async/Sync serial port p3 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.	This interrupt is cleared when data is read from the R_SERIAL3_REC_DATA register, or from the data_in field of R_SERIAL3_READ.
ser2_ready	13	This interrupt is set when asynchronous serial port p2 is ready to acquire new data for transmission. It should be masked when the DMA is used for data transfers.	This interrupt is cleared by writing new data to the R_SERIAL2_TR_DATA register, or to the data_out field of R_SERIAL2_CTRL.
ser2_data	12	When asynchronous serial port p2 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.	This interrupt is cleared when data is read from the R_SERIAL2_REC_DATA register, or from the data_in field of R_SERIAL2_READ.
ser1_ready	11	This interrupt is set when Async/Sync serial port p1 is ready to acquire new data for transmission. It should be masked when the DMA is used for data transfers.	This interrupt is cleared by writing new data to the R_SERIAL1_TR_DATA register, or to the data_out field of R_SERIAL1_CTRL.
ser1_data	10	When Async/Sync serial port p1 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.	This interrupt is cleared when data is read from the R_SERIAL1_REC_DATA register, or from the data_in field of R_SERIAL1_READ.
ser0_ready	9	This interrupt is set when asynchronous serial port p0 is ready to acquire new data for transmission. It should be masked when the DMA is used for data transfers.	This interrupt is cleared by writing new data to the R_SERIAL0_TR_DATA register, or to the data_out field of R_SERIAL0_CTRL.
ser0_data	8	When asynchronous serial port p0 is in use, this interrupt is set when data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.	This interrupt is cleared when data is read from the R_SERIAL0_REC_DATA register, or from the data_in field of R_SERIAL0_READ.

Table 17-16 Asynchronous and Synchronous Serial Port Interrupts

General Port PA Interrupts

Register sub-set 1 has eight fields for handling the active-high, level-triggered interrupts at general I/O port PA. They all have the internally generated vector number 0x2B.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
pa7	7	This interrupt is on bit 7 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 7 of general port PA.
pa6	6	This interrupt is on bit 6 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 6 of general port PA.
pa5	5	This interrupt is on bit 5 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 5 of general port PA.
pa4	4	This interrupt is on bit 4 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 4 of general port PA.
pa3	3	This interrupt is on bit 3 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 3 of general port PA.
pa2	2	This interrupt is on bit 2 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 2 of general port PA.
pa1	1	This interrupt is on bit 1 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 1 of general port PA.
pa0	0	This interrupt is on bit 0 of general port PA when the port is used for interrupt handling.	This interrupt is cleared in the external unit that is driving bit 0 of general port PA.

Table 17-17 General Port PA Interrupts

17.6.3 Interrupts in Register Sub-Set 2

Register sub-set 2 handles all the interrupts for the DMA channels. The masked interrupts are set in register R_IRQ_MASK2_SET and read in register R_IRQ_MASK2_RD. Register R_IRQ_READ2 contains the interrupt bits before they are individually masked. The asserted mask bits are cleared in register R_IRQ_MASK2_CLR.

Each interrupt field occupies the same bit number in all the registers of sub-set 2.

Sub-Channel Interrupts for DMA Channel 8

Register sub-set 2 has four fields for handling the descriptor interrupts of the four sub-channels in DMA channel 8.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
dma8_sub3_descr	23	This is the descriptor interrupt for DMA channel 8 sub-channel 3. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_descr bit in DMA register R_DMA_CH8_SUB3_CLR_INTR.
dma8_sub2_descr	22	This is the descriptor interrupt for DMA channel 8 sub-channel 2. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_descr bit in DMA register R_DMA_CH8_SUB2_CLR_INTR.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
dma8_sub1_descr	21	This is the descriptor interrupt for DMA channel 8 sub-channel 1. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_descr bit in DMA register R_DMA_CH8_SUB1_CLR_INTR.
dma8_sub0_descr	20	This is the descriptor interrupt for DMA channel 8 sub-channel 0. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_descr bit in DMA register R_DMA_CH8_SUB0_CLR_INTR.

Table 17-18 Sub-Channel Interrupts for DMA Channel 8

DMA End-of-Packet and Descriptor Interrupts

Register sub-set 2 handles all the end-of-packet and descriptor interrupts for the ten DMA channels.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
dma9_eop	19	This is the end-of-packet interrupt for DMA channel 9. It has the internally generated vector number 0x39.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH8_CLR_INTR.
dma9_descr	18	This is the descriptor interrupt for DMA channel 9. It has the internally generated vector number 0x39.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH8_CLR_INTR.
dma8_eop	17	This is the end-of-packet interrupt for DMA channel 8. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH8_CLR_INTR.
dma8_descr	16	This is the descriptor interrupt for DMA channel 8. It has the internally generated vector number 0x38.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH8_CLR_INTR.
dma7_eop	15	This is the end-of-packet interrupt for DMA channel 7. It has the internally generated vector number 0x37.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH7_CLR_INTR.
dma7_descr	14	This is the descriptor interrupt for DMA channel 7. It has the internally generated vector number 0x37.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH7_CLR_INTR.
dma6_eop	13	This is the end-of-packet interrupt for DMA channel 6. It has the internally generated vector number 0x36.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH6_CLR_INTR.
dma6_descr	12	This is the descriptor interrupt for DMA channel 6. It has the internally generated vector number 0x36.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH6_CLR_INTR.
dma5_eop	11	This is the end-of-packet interrupt for DMA channel 5. It has the internally generated vector number 0x35.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH5_CLR_INTR.
dma5_descr	10	This is the descriptor interrupt for DMA channel 5. It has the internally generated vector number 0x35.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH5_CLR_INTR.
dma4_eop	9	This is the end-of-packet interrupt for DMA channel 4. It has the internally generated vector number 0x34.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH4_CLR_INTR.
dma4_descr	8	This is the descriptor interrupt for DMA channel 4. It has the internally generated vector number 0x34.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH4_CLR_INTR.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
dma3_eop	7	This is the end-of-packet interrupt for DMA channel 3. It has the internally generated vector number 0x33.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH3_CLR_INTR.
dma3_descr	6	This is the descriptor interrupt for DMA channel 3. It has the internally generated vector number 0x33.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH3_CLR_INTR.
dma2_eop	5	This is the end-of-packet interrupt for DMA channel 2. It has the internally generated vector number 0x32.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH2_CLR_INTR.
dma2_descr	4	This is the descriptor interrupt for DMA channel 2. It has the internally generated vector number 0x32.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH2_CLR_INTR.
dma1_eop	3	This is the end-of-packet interrupt for DMA channel 1. It has the internally generated vector number 0x31.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH1_CLR_INTR.
dma1_descr	2	This is the descriptor interrupt for DMA channel 1. It has the internally generated vector number 0x31.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH1_CLR_INTR.
dma0_eop	1	This is the end-of-packet interrupt for DMA channel 0. It has the internally generated vector number 0x30.	This interrupt is cleared by asserting the clr_eop bit in register R_DMA_CH0_CLR_INTR.
dma0_descr	0	This is the descriptor interrupt for DMA channel 0. It has the internally generated vector number 0x30.	This interrupt is cleared by asserting the clr_descr bit in register R_DMA_CH0_CLR_INTR.

Table 17-19 DMA End-of-Packet and Descriptor Interrupts

17.6.4 Interrupts in the USB Register Set

Register sub-set R_USB_IRQ handles all the interrupts for the USB interface. The masked interrupts are set in register R_USB IRQ_MASK_SET and read in register R_USB_IRQ_MASK_RD. Register R_USB IRQ_READ contains the interrupt bits before they are individually masked. The asserted mask bits are cleared in register R_USB IRQ_MASK_CLR.

All USB interrupts have the internally generated vector number 0x3F. Each interrupt field occupies the same bit number in all the registers of this sub-set.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
iso_eof	13	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the isochronous endpoint DMA list.	This interrupt is cleared when the iso_eof bit in register R_USB_EPID_ATTN is read.
intr_eof	12	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the interrupt endpoint DMA list.	This interrupt is cleared when the intr_eof bit in register R_USB_EPID_ATTN is read.
iso_eot	11	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the isochronous endpoint DMA list, and the USB interface has finished the last transaction of the transfer.	This interrupt is cleared when the iso_eot bit in register R_USB_EPID_ATTN is read.

intr_eot	10	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the interrupt endpoint DMA list, and the USB interface has finished the last transaction of the transfer.	This interrupt is cleared when the intr_eot bit in register R_USB_EPID_ATTN is read.
ctl_eot	9	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the control endpoint DMA list, and the USB interface has finished the last transaction of the transfer.	This interrupt is cleared when the ctrl_eot bit in register R_USB_EPID_ATTN is read.
bulk_eot	8	This interrupt occurs when the DMA detects a set eof bit in an endpoint descriptor in the bulk endpoint DMA list, and the USB interface has finished the last transaction of the transfer.	This interrupt is cleared when bulk_eot bit in register R_USB_EPID_ATTN is read.
epid_attn	3	This interrupt is triggered whenever a significant endpoint event occurs.	This interrupt is cleared when register R_USB_EPID_ATTN is read.
sof	2	This interrupt is triggered each time the USB outputs a start-of-frame.	This interrupt is cleared when register R_USB_FM_NUMBER is read.
port_status	1	This interrupt signals a change in any of the configured port status registers.	This interrupt is cleared after all changed port status registers (R_USB_RH_PORT_STATUS_{12}) have been read.
ctl_status	0	This interrupt indicates a change of USB interface controller status.	This interrupt is cleared when register R_USB_STATUS is read.

Table 17-20 Interrupts in the USB Register Set

17.6.5 Vector Number Register Sub-Set

In this sub-set the composite masked interrupts are set in register R_VECT_MASK_SET and read in register R_VECT_MASK_RD. Register R_VECT_READ contains the composite interrupt bits before they are masked. The asserted composite mask bits are cleared in register R_VECT_MASK_CLR.

Each composite field occupies the same bit number in all the registers of the vector number sub-set.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
usb	31	This is the composed bit for interrupts from the USB. Vector number 0x3F.	This field is cleared by asserting the usb bit in register R_VECT_MASK_CLR.
dma9	25	This is the composed bit for interrupts from DMA channel 9. Vector number 0x39.	This field is cleared by asserting the dma9 bit in register R_VECT_MASK_CLR.
dma8	24	This is the composed bit for interrupts from DMA channel 8. Vector number 0x38.	This field is cleared by asserting the dma8 bit in register R_VECT_MASK_CLR.
dma7	23	This is the composed bit for interrupts from DMA channel 7. Vector number 0x37.	This field is cleared by asserting the dma7 bit in register R_VECT_MASK_CLR.
dma6	22	This is the composed bit for interrupts from DMA channel 6. Vector number 0x36.	This field is cleared by asserting the dma6 bit in register R_VECT_MASK_CLR.
dma5	21	This is the composed bit for interrupts from DMA channel 5. Vector number 0x35.	This field is cleared by asserting the dma5 bit in register R_VECT_MASK_CLR.
dma4	20	This is the composed bit for interrupts from DMA channel 4. Vector number 0x34.	This field is cleared by asserting the dma4 bit in register R_VECT_MASK_CLR.
dma3	19	This is the composed bit for interrupts from DMA channel 3. Vector number 0x33.	This field is cleared by asserting the dma3 bit in register R_VECT_MASK_CLR.

Name of Interrupt	Register Bit	Description of Interrupt	Interrupt Clearance
usb	31	This is the composed bit for interrupts from the USB. Vector number 0x3F.	This field is cleared by asserting the usb bit in register R_VECT_MASK_CLR.
dma2	18	This is the composed bit interrupts from DMA channel 2. Vector number 0x32.	This field is cleared by asserting the dma2 bit in register R_VECT_MASK_CLR.
dma1	17	This is the composed bit interrupts from DMA channel 1. Vector number 0x31.	This field is cleared by asserting the dma1 bit in register R_VECT_MASK_CLR.
dma0	16	This is the composed bit for interrupts from DMA channel 0. Vector number 0x30.	This field is cleared by asserting the dma0 bit in register R_VECT_MASK_CLR.
ext_dma1	13	This is the composed bit for interrupts from external DMA channel 1. Vector number 0x2D.	This field is cleared by asserting the ext_dma1 bit in register R_VECT_MASK_CLR.
ext_dma0	12	This is the composed bit for interrupts from external DMA channel 0. Vector number 0x2C.	This field is cleared by asserting the ext_dma0 bit in register R_VECT_MASK_CLR.
pa	11	This is the composed bit for interrupts from general port PA. Vector number 0x2B.	This field is cleared by asserting the pa bit in register R_VECT_MASK_CLR.
irq_intnr	10	This is the composed bit for interrupts from the $\overline{\text{irq}}$ pin, vector number 0x2A.	This field is cleared by asserting the irq_intnr bit in register R_VECT_MASK_CLR.
sw	9	This is the composed bit for software generated interrupts. Vector number 0x29.	This field is cleared by asserting the sw bit in register R_VECT_MASK_CLR.
serial	8	This is the composed bit for interrupts from the asynchronous and synchronous serial ports. Vector number 0x28.	This field is cleared by asserting the serial bit in register R_VECT_MASK_CLR.
snmp	7	This is the composed bit for Ethernet error and statistics counter interrupts. Vector number 0x27.	This field is cleared by asserting the snmp bit in register R_VECT_MASK_CLR.
network	6	This is the composed bit for interrupts from the network interface. Vector number 0x26.	This field is cleared by asserting the network bit in register R_VECT_MASK_CLR.
scsi1 par1	5	This is the composed bit for interrupts from SCSI-8 Port p1 and parallel port p1. Vector number 0x25.	This field is cleared by asserting bit 5 in register R_VECT_MASK_CLR.
scsi0 par0 ata mio	4	This is the composed bit for interrupts from SCSI-8 Port p0, parallel port p0, the ATA port and the shared RAM port. Vector number 0x24.	This field is cleared by asserting bit 4 in register R_VECT_MASK_CLR.
timer1	3	This is the composed bit for interrupts from timer 1. Vector number 0x23.	This field is cleared by asserting bit 3 in register R_VECT_MASK_CLR.
timer0	2	This is the composed bit for interrupts from timer 0. Vector number 0x22.	This field is cleared by asserting bit 2 in register R_VECT_MASK_CLR.
nmi	1	This is the composed bit for interrupts from the NMI. It cannot be masked and therefore the mask bit is always set to 1. Vector number 0x21.	This field cannot be cleared: it is always enabled.
some	0	This is the composed bit for interrupts (except NMI but including $\overline{\text{irq}}$ with an external vector number), that are active after the individual masks.	This field cannot be cleared: it is always enabled.

Table 17-21 Vector Number Register Sub-Set

17.7 External Maskable Interrupt with an External Vector Number

The external interrupt on the irq pin can be configured for an external vector number or an internally generated vector number.

To configure the external interrupt for an external vector number, the irq_ext_vector_nr field (bit 5) must be set in R_IRQ_MASK0_SET, and the irq_int_vector_nr field (bit 4) must be cleared in interrupt mask register R_IRQ_MASK0_CLR.

In the acknowledgement cycle for an interrupt with an external vector number, the inta output is asserted. In response, the external device supplies its vector number in the least significant byte on the data bus. Please refer to chapter 5.8 *External Interrupt Acknowledge* for more information.

17.8 Software Interrupts

As previously noted, the eight software interrupts with internally generated vector numbers are handled in register sub-set 1. They are set in register R_IRQ_MASK1_SET and cleared with their respective mask bits in register R_IRQ_MASK1_CLR, which actually constitutes the interrupts.

During normal program flow, the software may occasionally enter a critical region in which it cannot share resources. Typically this could occur when incoming data from an interface almost entirely fills a buffer, and usually requires that the interrupts are disabled. However ETRAX 100LX has a solution wherein the original interrupt routine can be enabled more often.

The software interrupts offer a simple and rapid response to hardware interrupts requiring actions that are extensive, but not so time-critical. When the original interrupt has executed the time-critical operation, it can set a software interrupt and return.

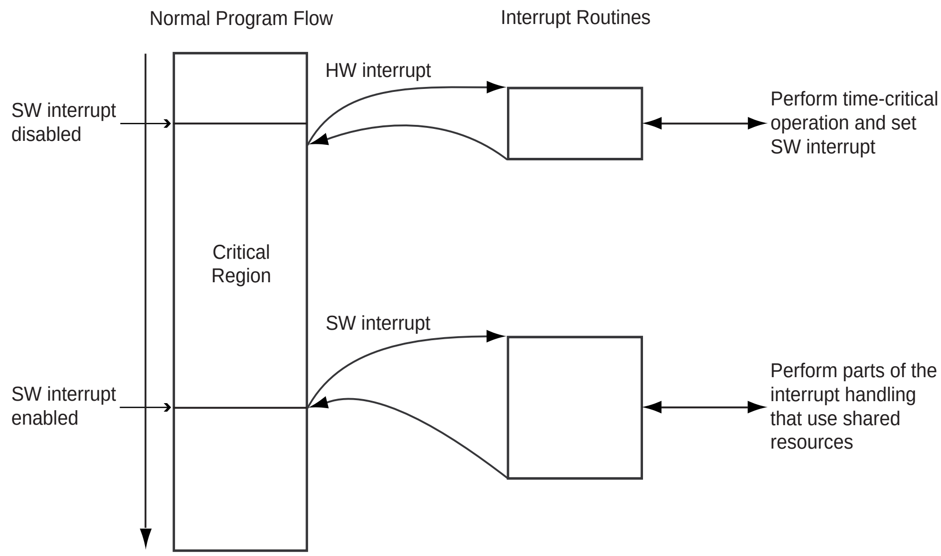


Figure 17-2 Software Generated Interrupt Routines

Figure 17-2 shows that, when the program enters a critical region, the software interrupt is disabled. The hardware interrupt then takes care of the time-critical operation and sets the software interrupt. As soon as the software interrupt is enabled, the software interrupt routine performs the more extensive action.

18 Internal Registers

This chapter contains detailed information about the internal registers in the ETRAX 100LX. The register descriptions are presented as appropriate sets, within which the content of each separate register is specified to bit level.

The description of each register is introduced by a table giving the identification (ID), offset value, register address, size of the register, read/write capability and initial value.

The bit allocation of each register is then presented in a table as follows:

Bit(s)	Name	Description	State/Range
bit no(s)	name of bit	Summary of the function of the bit and associated signals.	Significance of each bit state or range of bit field.

18.1 Conventions

18.1.1 Notation

All hexadecimal values, including addresses, are preceded by 0x. All other values are given in decimal notation.

18.1.2 Base Address

The internal registers are positioned at base address 0xB0000000 and upwards. Thus the offset value given for each register must be added to the base address to obtain the true address of the respective register.

18.2 Bus Interface Configuration Registers

18.2.1 R_WAITSTATES

Wait States Register, General Characteristics

ID of register	R_WAITSTATES	**Size**	32 bits
Offset	0x0	**Read/Write**	Write only
Address	0xB0000000	**Initial value**	0xFFFFFFFF

Bit Assignments of R_WAITSTATES

Bit(s)	Name	Description	State/Range
31- 30	pcs4_7_zw	This 2-bit field sets the number of wait states during bus turn-off for peripheral chip select signals $\overline{csp4}$ - $\overline{csp7}$. (note 1)	0 - 3

29 - 28	pcs4_7_ew	This 2-bit field sets the number of early wait states for peripheral chip select signals $\overline{csp4}$ - $\overline{csp7}$. (note 2)	0 - 3
27 - 24	pcs4_7_lw	This 4-bit field sets the number of late wait states for peripheral chip select signals $\overline{csp4}$ - $\overline{csp7}$. (note 3)	0 - 15
23 - 22	pcs0_3_zw	This 2-bit field sets the number of wait states during bus turn-off for peripheral chip select signals $\overline{csp0}$ - $\overline{csp3}$. (note 1)	0 - 3
21 - 20	pcs0_3_ew	This 2-bit field sets the number of early wait states for peripheral chip select signals $\overline{csp0}$ - $\overline{csp3}$. (note 2)	0 - 3
19 - 16	pcs0_3_lw	This 4-bit field sets the number of late wait states for peripheral chip select signals $\overline{csp0}$ - $\overline{csp3}$. (note 3)	0 - 15
15 - 14	sram_zw	This 2-bit field sets the number of wait states during bus turn-off for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$. (note 1)	0 - 3
13 - 12	sram_ew	This 2-bit field sets the number of early wait states for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$. (note 2)	0 - 3
11 - 8	sram_lw	This 4-bit field sets the number of late wait states for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$. (note 3)	0 - 15
7 - 6	flash_zw	This 2-bit field sets the number of wait states during bus turn-off for flash-PROM chip select signals $\overline{cse0}$ and $\overline{cse1}$. (note 1)	0 - 3
5 - 4	flash_ew	This 2-bit field sets the number of early wait states for flash-PROM chip select signals $\overline{cse0}$ and $\overline{cse1}$. (note 2)	0 - 3
3 - 0	flash_lw	This 4-bit field sets the number of late wait states for flash-PROM chip select signals $\overline{cse0}$ and $\overline{cse1}$. (note 3)	0 - 15

Note 1: The wait state is inserted after a burst to give the accessed unit sufficient time to turn off its drivers (read cycles), and to ensure sufficient data hold time (write cycles). The number of turn-off clock cycles is 1 + zw.

Note 2: The early wait state is inserted before $\overline{rd}$ or $\overline{wr}$ goes low in each bus cycle. The number of early clock cycles is ew. When ew is 0, $\overline{rd}$ will be low throughout the entire burst.

Note 3: The late wait state is inserted during the $\overline{rd}$ or $\overline{wr}$ low period. The number of late clock cycles is 2 + lw.

18.2.2 R_BUS_CONFIG

Bus Configuration Register, General Characteristics

ID of register	R_BUS_CONFIG	**Size**	32 bits
Offset	0x4	**Read/Write**	Write only
Address	0xB0000004	**Initial value**	0x0000000E or 0x0000000F (note 1)

Bit Assignments of R_BUS_CONFIG

Bit(s)	Name	Description	State/Range
31 - 10	Reserved	-	0
9	sram_type	This field selects the common write enable (cwe), or bytewise write enable (bwe) mode for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$.	0 = bwe 1 = cwe
8	dma_burst	This field selects 16 byte or 32 byte burst length for DMA transfers.	0 = burst32 1 = burst16
7	pcs4_7_wr	This field sets the write delay mode for peripheral chip select signals $\overline{csp4}$ - $\overline{csp7}$. (note 2)	0 = norm 1 = ext
6	pcs0_3_wr	This field sets the write delay mode for peripheral chip select signals $\overline{csp0}$ - $\overline{csp3}$. (note 2)	0 = norm 1 = ext
5	sram_wr	This field sets the write delay mode for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$. (note 2)	0 = norm 1 = ext
4	flash_wr	This field sets the write delay mode for EPROM/flash-PROM chip select signals $\overline{cse0}$ and $\overline{cse1}$. (note 2)	0 = norm 1 = ext
3	pcs4_7_bw	This field selects the bus width for peripheral chip select signals $\overline{csp4}$ - $\overline{csp7}$.	0 = bw16 1 = bw32
2	pcs0_3_bw	This field selects the bus width for peripheral chip select signals $\overline{csp0}$ - $\overline{csp3}$.	0 = bw16 1 = bw32
1	sram_bw	This field selects the bus width for SRAM chip select signals $\overline{csr0}$ and $\overline{csr1}$.	0 = bw16 1 = bw32
0	flash_bw	This field selects the bus width for EPROM/flash-PROM select signals $\overline{cse0}$ and $\overline{cse1}$.	0 = bw16 1 = bw32

Note 1: Bit 0 is set to the value of the bs0 pin at reset.

Note 2: If set to norm (0), the $\overline{wr}$ signals go high 0.5 clock cycles before the end of the bus cycle. If this bit is set to ext (1), the $\overline{wr}$ signals go high at the end of the bus cycle.

18.2.3 R_BUS_STATUS

Bus Status Register, General Characteristics

ID of register	R_BUS_STATUS	**Size**	32 bits
Offset	0x4	**Read/Write**	Read only
Address	0xB0000004	**Initial value**	See note 1.

Bit Assignments of R_BUS_STATUS

Bit(s)	Name	Description	State/Range
31 - 6	Reserved		0
5	pll_lock_tm	This bit shows the status of the timer that waits for PLL lock after reset. In normal operation, this bit should always be expired. When PLL bypass mode is enabled, the value changes from counting to expired after 32768 clock cycles.	0 = expired 1 = counting
4	both_faults	This bit is set if a single step bus fault and an MMU bus fault occur in the same CPU cycle. The bit is set to 0 by a CPU RBF instruction.	0 = no 1 = yes
3	bsen_	This bit shows if the outputs on the bus status pins $\overline{\text{bs3}}$ - $\overline{\text{bs0}}$ are enabled or disabled.	0 = enabled 1 = disabled
2 - 1	boot	This 2-bit field indicates the selected bootstrap method: - uncached starts at 0x80000002; - serial loads the program via serial port to cache; - network loads the program via network to cache; - parallel loads the program via parallel port to cache.	0 = uncached 1 = serial 2 = network 3 = parallel
0	flashw	This bit shows the initial width of the EPROM/flash-PROM banks 0 and 1, selected by $\overline{\text{cse0}}$ and $\overline{\text{cse1}}$ (address 0x00000000-0x07FFFFFF).	0 = bw16 1 = bw32

Note: Bit 4 is set to 0 after reset. Bits 3 to 0 are initiated through bus status pins at reset.

18.2.4 R_DRAM_TIMING

DRAM Timing Register, General Characteristics

ID of register	R_DRAM_TIMING	**Size**	32 bits
Offset	0x8	**Read/Write**	Write only
Address	0xB0000008	**Initial value**	0x0000FFFF

Bit Assignments of R_DRAM_TIMING

Bit(s)	**Name**	**Description**	**State/Range**
31	sdram	This bit enables or disables the SDRAM interface. When it is set, the SDRAM interface is enabled and the R_SDRAM_TIMING and R_SDRAM_CONFIG registers are used to configure the interface. (See the description of R_SDRAM_TIMING and R_SDRAM_CONFIG.)	0 = disable 1 = enable
30 - 16	Reserved	-	0
15 - 14	ref	This 2-bit field sets the DRAM refresh interval. For example e52us means that the DRAM will be refreshed at 52 µs intervals.	0 = e52us 1 = e13us 2 = e8700ns 3 = disable
13 - 12	rp	$\overline{ras}$ precharge waitstates. Wait states inserted after $\overline{ras}$ goes high, but before the row address is output (RAS-CAS cycle) or $\overline{cas}$ goes low (CBR refresh cycle). The number of precharge cycles is 0.5 + rp.	0 - 3
11 - 10	rs	Row address setup wait states. Wait states inserted between valid row address (RAS-CAS cycle) or $\overline{cas}$ low (CBR refresh cycle), and $\overline{ras}$ low. Number of setup cycles is 1 + rs.	0 - 3
9 - 8	rh	Row address hold time. The number of row address hold clock cycles is 1 + rh.	0 - 3
7	w	Write delay mode. If set to norm, the $\overline{wr}$ signals go high 0.5 clock cycles before the end of the bus cycle. If set to ext, the $\overline{wr}$ signals go high at the end of the bus cycle.	0 = norm 1 = ext
6	c	$\overline{cas}$ delay mode. If set to norm no extra delay is added, if set to ext the negative edge of $\overline{cas}$ is delayed 0.5 clock cycles. This does not affect the total time for the bus cycle.	0 = norm 1 = ext
5 - 4	cz	$\overline{cas}$ wait states during bus turn-off. Inserted after a burst to DRAM. Number of turn off cycles will be 1 + cz. In EDO mode, cz should only be set to 0 or 1, giving 1 + cz turn off clock cycles for write, and 3 + cz turn off clock cycles for read.	0 - 3
3 - 2	cp	$\overline{cas}$ precharge wait states. Inserted before $\overline{cas}$ goes low in each CAS only or RAS-CAS cycle. Also inserted while $\overline{cas}$ is low in CBR refresh cycles. Number of $\overline{cas}$ precharge cycles is 1 + cp.	0 - 3
1 - 0	cw	$\overline{cas}$ wait states. Inserted during $\overline{cas}$ low. Number of $\overline{cas}$ clock cycles i 1 + cw.	0 - 3

18.2.5 R_SDRAM_TIMING

SDRAM Timing Register, General Characteristics

ID of register	R_SDRAM_TIMING	**Size**	32 bits
Offset	0x8	**Read/Write**	Write only
Address	0xB0000008	**Initial value**	0x0000FFFF

Bit Assignments of R_SDRAM_TIMING

Bit(s)	Name	Description	State/Range
31	sdram	SDRAM enable. This bit must not be set before R_SDRAM_CONFIG is configured properly.	0 = disable 1 = enable
30 - 16	mrs_data	Data output on a15 - a1 during SDRAM mrs cycle.	
15 - 14	ref	This 2-bit field sets the SDRAM refresh interval. For example e52us means that the SDRAM will be refreshed at 52 µs intervals.	0 = e52us 1 = e13us 2 = e6500ns 3 = disable
13	ddr	Double Data Rate select. If this bit is set, the SDRAM interface functions as a DDR SDRAM interface.	0 = off 1 = on
12	clk100	SDRAM master clock select. If this bit is set, a 100MHz master clock will be used, if not a 50MHz master clock will be used.	0 = off 1 = on
11	ps	Power save select. If this bit is set, the SDRAM interface will enter power down mode after each refresh.	0 = off 1 = on
10 - 9	cmd	Initiate an SDRAM command cycle. The types available are “mode register set” (mrs), “refresh” (ref) or “precharge all” (pre) cycle. The ref and mrs command should always be preceded by a pre command. All commands must always be followed by a nop command e.g. pre, nop, mrs, nop. This field is used during initializing of the SDRAM modules.	0 = nop 1 = mrs 2 = ref 3 = pre
8	pde	Power down exit delay. Number of delay cycles from power down exit to new command will be pde + 1.	0 - 1
7 - 6	rc	Row cycle time. This is the auto refresh cycle time and will be rc + 6.	0 - 3
5 - 4	rp	$\overline{\text{ras}}$ precharge delay cycles. Number of delay cycles after precharge bank command will be rp + 1 in 50MHz mode and rp + 2 in 100MHz mode.	0 - 3
3 - 2	rcd	$\overline{\text{ras}}$ to $\overline{\text{cas}}$ delay. Number of delay cycles after activate bank command will be rcd + 1 in 50MHz mode and rcd + 2 in 100MHz mode.	0 - 3
1 - 0	cl	$\overline{\text{cas}}$ latency cycles. In 50MHz mode, the number of delay cycles from read bank command to valid read data will be cl + 1, with cl = 1 and cl = 2 as allowed values. In 100 MHz modes the number of cyles will be cl + 2, with cl = 0 and cl = 1 as allowed values. Thus $\overline{\text{cas}}$ latency can be varied from 2 to 3 SDRAM clock cycles. Note that cl = 3 is not allowed in either mode.	0 - 2

18.2.6 R_DRAM_CONFIG

DRAM Configuration Register, General Characteristics

ID of register	R_DRAM_CONFIG	**Size**	32 bits
Offset	0xC	**Read/Write**	Write only
Address	0xB000000C	**Initial value**	Unknown

Bit Assignments of R_DRAM_CONFIG

Bit(s)	Name	Description	State/Range
31	wmm1	Wide module mode for group 1. In wide module mode, all 8 $\overline{cas}$ outputs are used in each bank. The use of $\overline{casa3}$ - $\overline{casa0}$ or $\overline{casb3}$ - $\overline{casb0}$ is selected by the highest address bit in the selected column address range (ca1 field). The ca1 field should in this case be set to one bit higher than the highest column address bit to the DRAM. This mode is used with 64-bit wide DRAM modules that don't have separate $\overline{ras}$ strobes for the upper and lower half of the data bus.	0 = norm 1 = wmm
30	wmm0	Wide module mode for group 0. See wmm1 for description.	0 = norm 1 = wmm
29 - 27	sh1	This field gives the row address shift for group 1. The value given decides how many steps the address bits are shifted. Total shift is sh1 + 8. If, for example, sh1 equals 0 the internal address 29 - 9 is shifted down to pins A21 - A1 and if sh1 equals 7 the internal address 29 - 16 is shifted down to pins A14 - A1.	0 - 7
26 - 24	sh0	Row address shift for group 0. See sh1 for details.	0 - 7
23	w	DRAM bus width.	0 = bw16 1 = bw32
22	c	$\overline{cas}$ organization, byte- or bank-wise.	0 = byte 1 = bank
21	e	DRAM type select, fast page or EDO.	0 = fast 1 = edo
20 - 16	group_sel	Selects which address bit that will be used to select between the two groups of DRAM banks. Always group 0 (grp0), always group 1 (grp1) or use address bit to select between groups.	0 = grp0 1 = grp1 9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29

Bit Assignments of R_DRAM_CONFIG (continued)

Bit(s)	Name	Description	State/Range
15 - 13	ca1	Column address range for group 1. This selects how many bits above bit 8 that are used for the column address. If ca1 equals 0 up to and including address bit 8 are used, if ca1 equals 7 up to and including bit 15 are used.	0 - 7
12 - 8	bank23sel	Selects which address bit that will be used to select between the two banks in group 1. Always group 0 (grp0), always group 1 (grp1) or use address bit to select between groups.	0 = bank0 1 = bank1 9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29
7 - 5	ca0	Column address range for group 0. See ca1 for details.	0 - 7
4 - 0	bank01sel	Selects which address bit that will be used to select between the two banks in group 0. See bank23sel for details.	0 = bank0 1 = bank1 9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29

18.2.7 R_SDRAM_CONFIG

SDRAM Configuration Register, General Characteristics

ID of register	R_SDRAM_CONFIG	**Size**	32 bits
Offset	0xC	**Read/Write**	Write only
Address	0xB000000C	**Initial value**	Unknown

Bit Assignments of R_SDRAM_CONFIG

Bit(s)	Name	Description	State/Range
31	wmm1	Wide module mode for group 1. In wide module mode, all 8 dqm outputs are used in each group. The use of dqm7 - dqm4 or dqm3 - dqm0 is selected by the highest address bit in the selected column address range (ca1 field). The ca1 field should in this case be set to one bit higher than the highest column address bit to the SDRAM. This mode is used with 64-bit wide SDRAM modules that do not have separate chip selects for the upper and lower half of the data bus.	0 = norm 1 = wmm
30	wmm0	Wide module mode for group 0. See wmm1 for description.	0 = norm 1 = wmm
29 - 27	sh1	This field gives the row address shift for group 1. The value given decides how many steps the address bits are shifted. Total shift is sh1 + 8. If, for example, sh1 equals 0 the internal address 29 - 9 is shifted down to pins A21 - A1 and if sh1 equals 7 the internal address 29 - 16 is shifted down to pins A14 - A1.	0 - 7
26 - 24	sh0	Row address shift for group 0. See sh1 for details.	0 - 7
23	w	SDRAM bus width.	0 = bw16 1 = bw32
22	type1	This field selects 2 or 4 SDRAM banks for group 1.	0 = bank2 1 = bank4
21	type0	This field selects 2 or 4 SDRAM banks for group 0.	0 = bank2 1 = bank4
20 - 16	group_sel	Selects which address bit that will be used to select between the two groups of SDRAM banks. Always group 0 (grp0), always group 1 (grp1) or use address bit to select between groups.	0 = grp0 1 = grp1 9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29

Bit Assignments of R_SDRAM_CONFIG (continued)

Bit(s)	Name	Description	State/Range
15 - 13	ca1	Column address range for group 1. This selects how many bits above bit 8 that are used for the column address. If ca1 equals 0 up to and including address bit 8 are used, If ca1 equals 7 up to and including bit 15 are used.	0 - 7
12 - 8	bank_sel1	Selects which address bit that will be used to select between bank 0/1 in group 1. In 4-bank mode bank 2/3 will be controlled by the next higher order address bit.	9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29
7 - 5	ca0	Column address range for group 0. See ca1 for details.	0 - 7
4 - 0	bank_sel0	Selects which address bit that will be used to select between banks 0/1 in group 0. See bank_sel1 for details.	9 = bit9 10 = bit10 11 = bit11 12 = bit12 13 = bit13 14 = bit14 15 = bit15 16 = bit16 17 = bit17 18 = bit18 19 = bit19 20 = bit20 21 = bit21 22 = bit22 23 = bit23 24 = bit24 25 = bit25 26 = bit26 27 = bit27 28 = bit28 29 = bit29

Note: When using one group only, set register field group_sel to either grp0 or grp1. When using two groups, both groups must be configured equally as to row address shift and cas address range bits, i.e. both groups must have the same size.

18.3 External DMA Registers

18.3.1 R_EXT_DMA_0_CMD

External DMA Channel 0 Command Register, General Characteristics

ID of register	R_EXT_DMA_0_CMD	Size	32 bits
Offset	0x10	Read/Write	Write only
Address	0xB0000010	Initial value	0

Bit Assignments of R_EXT_DMA_0_CMD

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	cnt	This field enables/disables the transfer counter. If it is set (enable), the external DMA transfer will be stopped when trf_count reaches 0.	0 = disable 1 = enable
22	rqpol	Request polarity. Active high (ahigh) or active low (alow).	0 = ahigh 1 = alow
21	apol	Acknowledge polarity. Active high (ahigh) or active low (alow).	0 = ahigh 1 = alow
20	rq_ack	Request/acknowledge mode.	0 = burst 1 = handsh
19 - 18	wid	This field decides the width of the transfer, 8 bits (byte), 16 bits (word) or 32 bits (dword).	0 = byte 1 = word 2 = dword
17	dir	Direction.	0 = input 1 = output
16	run	Start/stop.	0 = stop 1 = start
15 - 0	trf_count	Counter for number of transfers. If cnt is set, the external DMA transfer will stop when trf_count has counted down to 0. trf_count = 0 gives 65536 transfers. trf_count is always counting, irrespective of the state of cnt.	0 - 65535

18.3.2 R_EXT_DMA_0_STAT

External DMA Channel 0 Status Register, General Characteristics

ID of register	R_EXT_DMA_0_STAT	**Size**	32 bits
Offset	0x10	**Read/Write**	Read only
Address	0xB0000010	**Initial value**	0

Bit Assignments of R_EXT_DMA_0_STAT

Bit(s)	Name	Description	State/Range
31 - 17	Reserved	-	
16	run	Start/stop.	0 = stop 1 = start
15 - 0	trf_count	Counter for number of transfers. The counter is initialized by writing to the tfr_count field of the R_EXT_DMA_0_CMD register, and is decremented by 1 for each DMA access. When the counter reaches 0, the run bit is cleared if the cnt bit in the R_EXT_DMA_0_CMD register is set.	0 - 65535

18.3.3 R_EXT_DMA_0_ADDR

External DMA Channel 0 Address Register, General Characteristics

ID of register	R_EXT_DMA_0_ADDR	**Size**	32 bits
Offset	0x14	**Read/Write**	Write only
Address	0xB0000014	**Initial value**	Unknown

Bit Assignments of R_EXT_DMA_0_ADDR

Bit(s)	Name	Description	State/Range
31 - 30	Reserved	Always 2.	2
29 - 2	ext0_addr	Bit 29 - 2 of the address to where the external DMA channel 0 accesses are mapped. Bit 30 and 1 - 0 of the address are always 0.	
1 - 0	Reserved	Always 0.	0

18.3.4 R_EXT_DMA_I_CMD

External DMA Channel 1 Command Register, General Characteristics

ID of register	R_EXT_DMA_1_CMD	**Size**	32 bits
Offset	0x18	**Read/Write**	Write only
Address	0xB0000018	**Initial value**	0

Bit Assignments of R_EXT_DMA_1_CMD

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	cnt	This field enables/disables the transfer counter. If it is set (enable), the external DMA transfer will be stopped when trf_count reaches 0.	0 = disable 1 = enable
22	rqpol	Request polarity, active high (ahigh) or active low (alow).	0 = ahigh 1 = alow
21	apol	Acknowledge polarity, active high (ahigh) or active low (alow).	0 = ahigh 1 = alow
20	rq_ack	Request/acknowledge mode.	0 = burst 1 = handsh
19 - 18	wid	This field decides the width of the transfer, 8 bits (byte), 16 bits (word) or 32 bits (dword).	0 = byte 1 = word 2 = dword
17	dir	Direction.	0 = input 1 = output
16	run	Start/stop.	0 = stop 1 = start
15 - 0	trf_count	Counter for number of transfers. If cnt is set, the external DMA transfer will stop when trf_count has counted down to 0. trf_count = 0 gives 65536 transfers. trf_count is always counting.	0 - 65535

18.3.5 R_EXT_DMA_I_STAT

External DMA Channel 1 Status Register, General Characteristics

ID of register	R_EXT_DMA_1_STAT	**Size**	32 bits
Offset	0x18	**Read/Write**	Read only
Address	0xB0000018	**Initial value**	0

Bit Assignments of R_EXT_DMA_1_STAT

Bit(s)	Name	Description	State/Range
31 - 17	Reserved	-	
16	run	Start/stop.	0 = stop 1 = start
15 - 0	trf_count	Counter for number of transfers. The counter is initialized by writing to the tfr_count field of the R_EXT_DMA_1_CMD register, and is decremented by 1 for each DMA access. When the counter reaches 0, the run bit is cleared if the cnt bit in the R_EXT_DMA_1_CMD register is set.	0 - 65535

18.3.6 R_EXT_DMA_I_ADDR

External DMA Channel 1 Address Register, General Characteristics

ID of register	R_EXT_DMA_1_ADDR	**Size**	32 bits
Offset	0x1C	**Read/Write**	Write only
Address	0xB000001C	**Initial value**	Unknown

Bit Assignments of R_EXT_DMA_1_ADDR

Bit(s)	Name	Description	State/Range
31 - 30	Reserved	Always 2.	2
29 - 2	ext0_addr	Bit 29 - 2 of the address to where the external DMA channel 0 accesses are mapped. Bit 30 and 1 - 0 of the address are always 0.	
1 - 0	Reserved	Always 0.	0

18.4 Timer Registers

18.4.1 R_TIMER_CTRL

Timer Control Register, General Characteristics

ID of register	R_TIMER_CTRL	**Size**	32 bits
Offset	0x20	**Read/Write**	Write only
Address	0xB0000020	**Initial value**	Unknown

Bit Assignments of R_TIMER_CTRL

Bit(s)	Name	Description	State/Range
31 - 24	timerdiv1	This field contains the divide factor for timer 1. The usable range is 2 - 256. A divide factor of 256 is achieved by setting the field to 0. If the timer is used as an interval timer, i.e. generating a fixed interrupt frequency, the frequency will be the clksel1 frequency divided by the divide factor.	0 - 255
23 - 16	timerdiv0	This field contains the divide factor for timer 0. See timerdiv1.	0 - 255
15	presc_timer1	If this bit is set, the programmable clock divider is used to clock timer 1, thus overriding all settings to clksel1.	0 = normal 1 = prescale
14	i1	This bit is the interrupt acknowledge for timer 1. An interrupt is acknowledged by setting this bit to clr. The bit value is not saved, but reverts to nop once the interrupt has been cleared.	0 = nop 1 = clr
13 - 12	tm1	This field defines the operation of timer 1: if the value is stop_ld, the timer stops and loads the divide factor, if the value is freeze, the timer stops and preserves current count value and if it is run, the timer starts. reserved should not be used.	0 = stop_ld 1 = freeze 2 = run 3 = reserved
11 - 8	clksel1	Clock select for timer 1. cascade0 cascades this timer with timer 0.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = cascade0
7	presc_ext	This bit is used to select the clock for flexible in the clksel0 field. If it is set to prescale the programmable clock divider clock is selected. If it set to external the external clock is selected.	0 = prescale 1 = external
6	i0	This bit is the interrupt acknowledge for timer 0. See i1.	0 = nop 1 = clr

Bit Assignments of R_TIMER_CTRL (continued)

Bit(s)	Name	Description	State/Range
5 - 4	tm0	This field defines the operation of timer 0. See tm1.	0 = stop_ld 1 = freeze 2 = run 3 = reserved
3 - 0	clksel0	Clock select for timer 0.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = flexible

18.4.2 R_TIMER_DATA

Timer Data Register, General Characteristics

ID of register	R_TIMER_DATA	**Size**	32 bits
Offset	0x20	**Read/Write**	Read only
Address	0xB0000020	**Initial value**	Bit 31 - 16 unknown, bit 15 - 0 set to 0 at reset.

Bit Assignments of R_TIMER_DATA

Bit(s)	Name	Description	State/Range
31 - 24	timer1	Current count value for timer1. Note that this is a down counter that starts at the timerdiv1 value and then counts down to 1.	0-255
23 - 16	timer0	Current count value for timer0. Note that this is a down counter that starts at the timerdiv0 value and then counts down to 1.	0-255
15 - 8	clkdiv_high	High byte of clock divider. The bits in this field each toggle at a specific frequency according to: bit 8 - 38.4 kHz, bit 9 - 19.2 kHz, bit 10 - 9.6 kHz, bit 11 - 4.8 kHz, bit 12 - 2.4 kHz, bit 13 - 1.2 kHz, bit 14 - 600 Hz, bit 15 - 300 Hz.	
7 - 0	clkdiv_low	Low byte of clock divider. The bits in this field each toggle at a specific frequency according to: bit 0 - 7.3728 MHz, bit 1 - 3.6864 MHz, bit 2 - 1.8432 MHz, bit 3 - 921.6 kHz, bit 4 - 460.8 kHz, bit 5 - 230.4 kHz, bit 6 - 115.2 kHz, bit 7 - 57.6 kHz.	

18.4.3 R_TIMER01_DATA

Timer 01 Data Register

ID of register	R_TIMER01_DATA	**Size**	16 bits
Offset	0x22	**Read/Write**	Read only
Address	0xB0000022	**Initial value**	Unknown

Bit Assignments of R_TIMER01_DATA

Bit(s)	Name	Description	State/Range
15 - 0	count	The combination of the values in fields timer1 and timer0 in R_TIMER_DATA (timer1[7:0], timer0[7:0]). Typically used when the timers are cascaded.	0-65535

Note: This is a 16 bit wide register that is part of register R_TIMER_DATA.

18.4.4 R_TIMER0_DATA

Timer 0 Data Register, General Characteristics

ID of register	R_TIMER0_DATA	**Size**	8 bits
Offset	0x22	**Read/Write**	Read only
Address	0xB0000022	**Initial value**	Unknown

Bit Assignments of R_TIMER0_DATA

Bit(s)	Name	Description	State/Range
7 - 0	count	Current count value for timer0. Note that this is a down counter that starts at the timerdiv0 value and then counts down to 1.	0-255

Note: This is a 8 bit wide register which is part of register R_TIMER_DATA.

18.4.5 R_TIMER1_DATA

Timer 1 Data Register, General Characteristics

ID of register	R_TIMER1_DATA	**Size**	8 bits
Offset	0x23	**Read/Write**	Read only
Address	0xB0000023	**Initial value**	Unknown

Bit Assignments of R_TIMER1_DATA

Bit(s)	Name	Description	State/Range
7 - 0	count	Current count value for timer1. Note that this is a down counter that starts at the timerdiv1 value and then counts down to 1.	0-255

Note: This is a 8 bit wide register that is part of register R_TIMER_DATA.

18.4.6 R_WATCHDOG

Watchdog Register, General Characteristics

ID of register	R_WATCHDOG	**Size**	32 bits
Offset	0x24	**Read/Write**	Write only
Address	0xB0000024	**Initial value**	0

Bit Assignments of R_WATCHDOG

Bit(s)	Name	Description	State/Range
31 - 4	Reserved	-	0
3 - 1	key	Key value for the watchdog.	0 - 7
0	enable	Watchdog start/stop. If started, the watchdog timer generates an NMI if it is not restarted or stopped within 0.1 s. If it still is not restarted or stopped after an additional 3.3 ms, it resets the chip. Restart and stop only take effect if the new key value matches the inverse of the previously set key value.	0 = stop 1 = start

18.4.7 R_CLOCK_PRESCALE

Clock Prescale Register, General Characteristics

ID of register	R_CLOCK_PRESCALE	**Size**	32 bits
Offset	0xF0	**Read/Write**	Write only
Address	0xB00000F0	**Initial value**	Unknown

Bit Assignments of R_CLOCK_PRESCALE

Bit(s)	Name	Description	State/Range
31 - 16	ser_presc	This fields gives the divide factor for serial clock prescaling. It is used when another baud rate than those predefined in the serial port control registers is needed for the asynchronous serial ports. The usable range is 2 - 65536. A divide factor of 65536 is achieved by setting the field to 0. The resulting baud rate equals 3.125MHz divided by the divide factor. The prescaling starts when the field is written.	0, 2-65535
15 - 0	tim_presc	This field gives the divide factor for timer clock prescaling. It is used when another frequency than those predefined in R_TIMER_CTRL is needed for the internal timers.The usable range is 2 - 65536. A divide factor of 65536 is achieved by setting the field to 0. The generated frequency equals 25MHz divided by the divide factor. Thus the highest available frequency is 12.5 MHz and lowest available frequency is 381.5Hz. The prescaling starts when the field is written.	0, 2-65535

18.4.8 R_TIMER_PRESCALE

Timer Prescale Register, General Characteristics

ID of register	R_TIMER_PRESCALE	**Size**	16 bits
Offset	0xF0	**Read/Write**	Write only
Address	0xB00000F0	**Initial value**	Unknown

Bit Assignments of R_TIMER_PRESCALE

Bit(s)	Name	Description	State/Range
15 - 0	tim_presc	This field gives the divide factor for timer clock prescaling. It is used when another frequency than those predefined in R_TIMER_CTRL is needed for the internal timers.The usable range is 2 - 65536. A divide factor of 65536 is achieved by setting the field to 0. The generated frequency equals 25MHz divided by the divide factor. Thus the highest available frequency is 12.5 MHz and lowest available frequency is 381.5Hz. The prescaling starts when the field is written.	0, 2-65535

Note: This is a 16 bit wide register that is part of register R_CLOCK_PRESCALE.

18.4.9 R_PRESCALE_STATUS

Prescale Status Register, General Characteristics

ID of register	R_PRESCALE_STATUS	**Size**	32 bits
Offset	0xF0	**Read/Write**	Read only
Address	0xB00000F0	**Initial value**	Unknown

Bit Assignments of R_PRESCALE_STATUS

Bit(s)	Name	Description	State/Range
31 - 16	ser_status	Contains the current count value of the serial divide factor.	0-65535
15 - 0	tim_status	Contains the current count value of the timer divide factor.	0-65535

18.4.10 R_TIM_PRESC_STATUS

Timer Prescale Status Register, General Characteristics

ID of register	R_TIM_PRESC_STATUS	**Size**	16 bits
Offset	0xF0	**Read/Write**	Read only
Address	0xB00000F0	**Initial value**	Unknown

Bit Assignments of R_TIM_PRESC_STATUS

Bit(s)	Name	Description	State/Range
15 - 0	tim_status	Contains the current count value of the timer divide factor.	0-65535

Note: This is a 16 bit wide register that is part of register R_CLOCK_PRESCALE.

18.5 Shared RAM Interface Registers

To initiate the shared RAM interface, first write to R_GEN_CONFIG, then to R_SHARED_RAM_ADDR, and finally write to R_SHARED_RAM_CONFIG with ONLY the enable bit set.

18.5.1 R_SHARED_RAM_CONFIG

Shared RAM Configuration Register, General Characteristics

ID of register	R_SHARED_RAM_CONFIG	**Size**	32 bits
Offset	0x40	**Read/Write**	Write only
Address	0xB0000040	**Initial value**	0

Bit Assignments of R_SHARED_RAM_CONFIG

Bit(s)	Name	Description	State/Range
31 - 4	Reserved	-	0
3	width	This field decides the shared RAM interface width, byte (8 bits) or word (16 bits).	0 = byte 1 = word
2	enable	If this field is set the shared RAM interface is enabled.	0 = no 1 = yes
1	pint	If this field is set, a peripheral interrupt is generated. int generates a 600ns active low pulse on the $\overline{\text{pr\_int}}$ output. The bit value is not saved, but reverts to nop once the pulse has been generated.	0 = nop 1 = int
0	clri	This field clears the interrupt from the $\overline{\text{intio}}$ input. The bit value is not saved, but reverts to nop once the interrupt has been cleared.	0 = nop 1 = clr

18.5.2 R_SHARED_RAM_ADDR

Shared RAM Address Register, General Characteristics

ID of register	R_SHARED_RAM_ADDR	Size	32 bits
Offset	0x44	Read/Write	Write only
Address	0xB0000044	Initial value	Unknown

Bit Assignments of R_SHARED_RAM_ADDR

Bit(s)	Name	Description	State/Range
31 - 30	Reserved	This part of the Shared RAM address is always 2 (10 binary), to address non cached, non DRAM area.	2
29 - 8	base_addr	This field sets bit 29-8 of the base address for the shared RAM area. Bit 30 and 7 - 0 of the base address are always 0. Shared RAM accesses are always non-cacheable.	
7 - 0	Reserved	-	0

18.6 General Configuration Registers

18.6.1 R_GEN_CONFIG

General Configuration Register, General Characteristics

ID of register	R_GEN_CONFIG	**Size**	32 bits
Offset	0x2C	**Read/Write**	Write only
Address	0xB000002C	**Initial value**	0

Bit Assignments of R_GEN_CONFIG

Bit(s)	Name	Description	State/Range
31	par_w	If this field is set, parallel port wide (ecp_16) is selected. (note 1)	0 = disable 1 = select
30	usb2	This field selects USB port 2.	0 = disable 1 = select
29	usb1	This field selects USB port 1.	0 = disable 1 = select
28	Reserved	-	0
27	g24dir	This field selects direction for pin g24 (if configured).	0 = in 1 = out
26	g16_23dir	This field selects direction for pins g16 - 23 (if configured).	0 = in 1 = out
25	g8_15dir	This field selects direction for pins g8 -15 (if configured).	0 = in 1 = out
24	g0dir	This field selects direction for pin g0 (if configured).	0 = in 1 = out
23	dma9	This field decides the configuration for DMA channel 9. It can be connected to either USB or serial port 1. (note 2)	0 = usb 1 = serial1
22	dma8	This field decides the configuration for DMA channel 8. It can be connected to either USB or serial port 1. (note 2)	0 = usb 1 = serial1
21 - 20	dma7	This field decides the configuration for DMA channel 7. It is either unused, connected to serial port 0, to external DMA 1 or to internal DMA 6 (memory to memory DMA).	0 = unused 1 = serial0 2 = extdma1 3 = intdma6
19 - 18	dma6	This field decides the configuration for DMA channel 6. It is either unused, connected to serial port 0, to external DMA 1 or to internal DMA 7 (memory to memory DMA).	0 = unused 1 = serial0 2 = extdma1 3 = intdma7
17 - 16	dma5	This field decides the configuration for DMA channel 5. It can be connected to parallel port 1, SCSI 1, serial port 3 or external DMA 0. (note 2)	0 = par1 1 = scsi1 2 = serial3 3 = extdma0
15 - 14	dma4	This field decides the configuration for DMA channel 4. It can be connected to parallel port 1, SCSI 1, serial port 3 or external DMA 0. (note 2)	0 = par1 1 = scsi1 2 = serial3 3 = extdma0

Bit Assignments of R_GEN_CONFIG (continued)

Bit(s)	Name	Description	State/Range
13 - 12	dma3	This field decides the configuration of DMA channel 3. It can be connected to parallel port 0, SCSI 0, serial port 2 or ATA. (note 2)	0 = par0 1 = scsi0 2 = serial2 3 = ata
11 - 10	dma2	This field decides the configuration of DMA channel 2. It can be connected to parallel port 0, SCSI 0, serial port 2 or ATA. (note 2)	0 = par0 1 = scsi0 2 = serial2 3 = ata
9	mio_w	This field selects shared RAM wide (16 bits). The mio bit has to be set (see below). (note 3)	0 = disable 1 = select
8	ser3	This field selects serial port 3. (note 3)	0 = disable 1 = select
7	par1	This field selects parallel port 1. (note 3)	0 = disable 1 = select
6	scsi0w	This field selects wide (16 bits) scsi for scsi port 0. Bit scsi0 has to be set (see below). (note 3)	1 = select 0 = disable
5	scsi1	This field selects scsi port 1 (8 bits). (note 3)	0 = disable 1 = select
4	mio	This field selects shared RAM interface. (note 3)	0 = disable 1 = select
3	ser2	This field selects serial port 2. (note 3)	0 = disable 1 = select
2	par0	This field selects parallel port 0. (note 3)	0 = disable 1 = select
1	ata	This field selects ata. (note 3)	0 = disable 1 = select
0	scsi0	This field select scsi 0. If scsi0w is also set, scsi wide is selected. (note 3)	0 = disable 1 = select

Note 1: Both the wide field in R_PAR0_CONFIG and the par0 field in R_GEN_CONFIG must also be set, in order to activate the wide-mode in the parallel port.

Note 2: When a DMA channel is not used, it should be connected to an unused device. For example if DMA channel 5 and parallel port p1 are not in use, connect DMA channel 5 to parallel port p1.

Note 3: Only some combinations of bits, that are set, are meaningful. For example parallel port p0 and SCSI-8 port p0 can not be used simultaneously. See chapter 19.10 *Multiplexed Interfaces* for details.

Note 4: There must be a delay of >120 ns between writes to bits 23 to 0. Writing to bits 23 to 8 without writing to bits 7 to 0 has an undefined result.

18.6.2 R_GEN_CONFIG_II

General Configuration register II, General Characteristics

ID of register	R_GEN_CONFIG_II	**Size**	32 bits
Offset	0x34	**Read/Write**	Write only
Address	0xB0000034	**Initial value**	0

Bit Assignments of R_GEN_CONFIG_II

Bit(s)	Name	Description	State/Range
31 - 7	Reserved	-	0
6	sermode3	This bit decides if serial port 3 is used in asynchronous or in synchronous mode. If this field is set to sync, serial port 3 is in the synchronous mode.	0 = async 1 = sync
5	Reserved	-	0
4	sermode1	This bit decides if serial port 1 is used in asynchronous or in synchronous mode. If this field is set to sync, serial port 1 is in the synchronous mode.	0 = async 1 = sync
3	Reserved	-	0
2	ext_clk	If this bit is set, the external clock is selected for the serial ports and the timers.	0 = disable 1 = select
1 - 0	Reserved	-	0

18.6.3 R_PORT_G_DATA

General Port Data Register, General Characteristics

ID of register	R_PORT_G_DATA	**Size**	32 bits
Offset	0x28	**Read/Write**	Read/Write
Address	0xB0000028	**Initial value**	0

Bit Assignments of R_PORT_G_DATA

Bit(s)	Name	Description	State/Range
31 - 0	data	Through this register data can be read from and written to the IO pins g0-31. The different IO-devices are multiplexed on these pins. The availability of the pins varies with the setting of R_GEN_CONFIG bits 0 - 9.	

18.7 General Port Configuration Registers

18.7.1 R_PORT_PA_SET

General Port PA Set Register, General Characteristics

ID of register	R_PORT_PA_SET	**Size**	32 bits
Offset	0x30	**Read/Write**	Write only
Address	0xB0000030	**Initial value**	0

Bit Assignments of R_PORT_PA_SET

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	0
15	dir7	This bit sets the direction (input or output) of bit 7 in general port PA.	0 = input 1 = output
14	dir6	This bit sets the direction (input or output) of bit 6 in general port PA.	0 = input 1 = output
13	dir5	This bit sets the direction (input or output) of bit 5 in general port PA.	0 = input 1 = output
12	dir4	This bit sets the direction (input or output) of bit 4 in general port PA.	0 = input 1 = output
11	dir3	This bit sets the direction (input or output) of bit 3 in general port PA.	0 = input 1 = output
10	dir2	This bit sets the direction (input or output) of bit 2 in general port PA.	0 = input 1 = output
9	dir1	This bit sets the direction (input or output) of bit 1 in general port PA.	0 = input 1 = output
8	dir0	This bit sets the direction (input or output) of bit 0 in general port PA.	0 = input 1 = output
7 - 0	data_out	Data byte to general port PA.	0 - 255

18.7.2 R_PORT_PA_DATA

General Port PA Data Register General Characteristics

ID of register	R_PORT_PA_DATA	**Size**	8 bits
Offset	0x30	**Read/Write**	Write only
Address	0xB0000030	**Initial value**	0

Bit Assignments of R_PORT_PA_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Data byte to general port PA.	0 - 255

Note: This is an 8 bit wide register that is part of register R_PORT_PA_SET.

18.7.3 R_PORT_PA_DIR

General Port PA Direction Register, General Characteristics

ID of register	R_PORT_PA_DIR	**Size**	8 bits
Offset	0x31	**Read/Write**	Write only
Address	0xB0000031	**Initial value**	0

Bit Assignments of R_PORT_PA_DIR

Bit(s)	Name	Description	State/Range
7	dir7	This bit sets the direction (input or output) of bit 7 in general port PA.	0 = input 1 = output
6	dir6	This bit sets the direction (input or output) of bit 6 in general port PA.	0 = input 1 = output
5	dir5	This bit sets the direction (input or output) of bit 5 in general port PA.	0 = input 1 = output
4	dir4	This bit sets the direction (input or output) of bit 4 in general port PA.	0 = input 1 = output
3	dir3	This bit sets the direction (input or output) of bit 3 in general port PA.	0 = input 1 = output
2	dir2	This bit sets the direction (input or output) of bit 2 in general port PA.	0 = input 1 = output
1	dir1	This bit sets the direction (input or output) of bit 1 in general port PA.	0 = input 1 = output
0	dir0	This bit sets the direction (input or output) of bit 0 in general port PA.	0 = input 1 = output

Note: This is an 8 bit wide register that is part of register R_PORT_PA_SET.

18.7.4 R_PORT_PA_READ

General Port PA Read Register, General Characteristics

ID of register	R_PORT_PA_READ	**Size**	32 bits
Offset	0x30	**Read/Write**	Read only
Address	0xB0000030	**Initial value**	Unknown

Bit Assignments of R_PORT_PA_READ

Bit(s)	Name	Description	State/Range
31 - 8	Reserved	-	
7 - 0	data_in	Data byte from general port PA.	0 - 255

18.7.5 R_PORT_PB_SET

General Port PB Set Register, General Characteristics

ID of register	R_PORT_PB_SET	**Size**	32 bits
Offset	0x38	**Read/Write**	Write only
Address	0xB0000038	**Initial value**	0

Bit Assignments of R_PORT_PB_SET

Bit(s)	Name	Description	State/Range
31 - 30	Reserved	-	0
29	syncser3	If this bit is set, pb7 is used for synchronous serial port p3, which in some modes needs an extra signal. The setting of the syncser3 field overrides that of the cs7 and dir7 fields. The setting of the syncser3 field is ignored if the scsi1 field is set to enph.	0 = port_cs 1 = ss3extra
28	syncser1	If this bit is set, pb4 is used for synchronous serial port p1, which in some modes needs an extra signal. The setting of the syncser1 field overrides that of the cs4 and dir4 fields. The setting of the syncser1 field is ignored if the scsi0 field is set to enph.	0 = port_cs 1 = ss1extra
27	i2c_en	This bit enables or disables the I2C mode at pb1 and pb0.	0 = off 1 = on
26	i2c_d	This bit is the I2C output data bit. When the I2C mode is selected (field i2c_en = on) and the output is enabled (field i2c_oe_ = enable), i2c_d is output on pb0. The i2c_d field is ignored if the I2C mode is not enabled (i2c_en = off). (note)	0 - 1
25	i2c_clk	This bit is the I2C clock signal. When the I2C mode is selected (field i2c_en = on), i2c_clk is output on pb1. The i2c_clk field is ignored if the I2C mode is not enabled (i2c_en = off).	0 - 1
24	i2c_oe_	This bit is the output enable for the I2C mode. When the I2C mode is selected (field i2c_en = on), this field enables or disables the output at pb0. The i2c_oe_ field is ignored if the I2C mode is not enabled (i2c_en = off).	0 = enable 1 = disable
23	cs7	This bit determines whether pb7 is a general I/O or peripheral chip-select signal. The setting of the cs7 field is ignored if the scsi1 field is set to enph or if the syncser3 field is set to ss3extra.	0 = port 1 = cs
22	cs6	This bit determines whether pb6 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
21	cs5	This bit determines whether pb5 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
20	cs4	This bit determines whether pb4 is a general I/O or peripheral chip-select signal. The setting of the cs4 field is ignored if the scsi0 field is set to enph or if the syncser1 field is set to ss1extra.	0 = port 1 = cs
19	cs3	This bit determines whether pb3 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
18	cs2	This bit determines whether pb2 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs

Bit Assignments of R_PORT_PB_SET (continued)

Bit(s)	Name	Description	State/Range
17	scsi1	This bit determines whether signal $\overline{\text{s1enph}}$ at port SCSI-8 p1 is output on pb7. In SCSI-W mode, the scsi1 field determines whether signal s0enloid is output on pb7. The setting of the scsi1 field overrides that of the cs7 and dir7 fields, as well as the syncser3 field.	0 = port_cs 1 = enph
16	scsi0	This bit determines whether signal $\overline{\text{s0enph}}$ at port SCSI-8 p0 is output on pb4. The setting of the scsi0 field overrides that of the cs4 and dir4 fields, as well as the syncser1 field.	0 = port_cs 1 = enph
15	dir7	This bit sets the direction (input or output) of pb7. The dir7 field is ignored if the scsi1 field is set to enph, if the syncser3 field is set to ss3extra or if the cs7 field is set to cs.	0 = input 1 = output
14	dir6	This bit sets the direction (input or output) of pb6. The dir6 field is ignored if the cs6 field is set to cs.	0 = input 1 = output
13	dir5	This bit sets the direction (input or output) of pb5. The dir5 field is ignored if the cs5 field is set to cs.	0 = input 1 = output
12	dir4	This bit sets the direction (input or output) of pb4. The dir4 field is ignored if the scsi0 field is set to enph, if the syncser1 field is set to ss1extra or if the cs4 field is set to cs.	0 = input 1 = output
11	dir3	This bit sets the direction (input or output) of pb3. The dir3 field is ignored if the cs3 field is set to cs.	0 = input 1 = output
10	dir2	This bit sets the direction (input or output) of pb2. The dir2 field is ignored if the cs2 field is set to cs.	0 = input 1 = output
9	dir1	This bit sets the direction (input or output) of pb1. The dir1 field is ignored if the i2c_en field is set to on.	0 = input 1 = output
8	dir0	This bit sets the direction (input or output) of pb0. The dir0 field is ignored if the i2c_en field is set to on.	0 = input 1 = output
7 - 0	data_out	Output data byte to pins that are configured for general I/O purposes at port PB.	0 - 255

Note: The I2C data bit should normally be an open collector output. Open collector behavior is achieved by setting the i2c_d bit to 0 and using the i2c_oe_ field to control the data output.

18.7.6 R_PORT_PB_DATA

General Port PB Data Register, General Characteristics

ID of register	R_PORT_PB_DATA	**Size**	8 bits
Offset	0x38	**Read/Write**	Write only
Address	0xB0000038	**Initial value**	0

Bit Assignments of R_PORT_PB_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Output data byte to pins that are configured for general I/O purposes at port PB.	0 - 255

Note: This is an 8 bit wide register that is part of register R_PORT_PB_SET.

18.7.7 R_PORT_PB_DIR

General Port PB Direction Register, General Characteristics

ID of register	R_PORT_PB_DIR	**Size**	8 bits
Offset	0x39	**Read/Write**	Write only
Address	0xB0000039	**Initial value**	0

Bit Assignments of R_PORT_PB_DIR

Bit(s)	Name	Description	State/Range
7	dir7	This bit sets the direction (input or output) of pb7. The dir7 field is ignored if the scsi1 field in R_PORT_PB_SET is set to enph, if the syncser3 field in R_PORT_PB_SET is set to ss3extra or if the cs7 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
6	dir6	This bit sets the direction (input or output) of pb6. The dir6 field is ignored if the cs6 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
5	dir5	This bit sets the direction (input or output) of pb5. The dir5 field is ignored if the cs5 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
4	dir4	This bit sets the direction (input or output) of pb4. The dir4 field is ignored if the scsi0 field in R_PORT_PB_SET is set to enph, if the syncser1 field in R_PORT_PB_SET is set to ss1extra or if the cs4 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
3	dir3	This bit sets the direction (input or output) of pb3. The dir3 field is ignored if the cs3 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
2	dir2	This bit sets the direction (input or output) of pb2. The dir2 field is ignored if the cs2 field in R_PORT_PB_SET is set to cs.	0 = input 1 = output
1	dir1	This bit sets the direction (input or output) of pb1. The dir1 field is ignored if the i2c_en field in R_PORT_PB_SET is set to on.	0 = input 1 = output
0	dir0	This bit sets the direction (input or output) of pb0. The dir0 field is ignored if the i2c_en field in R_PORT_PB_SET is set to on.	0 = input 1 = output

Note: This is an 8-bit wide register that is part of register R_PORT_PB_SET.

18.7.8 R_PORT_PB_CONFIG

General Port PB Configuration Register, General Characteristics

ID of register	R_PORT_PB_CONFIG	**Size**	8 bits
Offset	0x3A	**Read/Write**	Write only
Address	0xB000003A	**Initial value**	0

Bit Assignments of R_PORT_PB_CONFIG

Bit(s)	Name	Description	State/Range
7	cs7	This bit determines whether pb7 is a general I/O or peripheral chip-select signal. The setting of the cs7 field is ignored if the scsi1 field in R_PORT_PB_SET is set to enph or if the syncser3 field in R_PORT_PB_SET is set to ss3extra.	0 = port 1 = cs
6	cs6	This bit determines whether pb6 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
5	cs5	This bit determines whether pb5 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
4	cs4	This bit determines whether pb4 is a general I/O or peripheral chip-select signal. The setting of the cs4 field is ignored if the scsi0 field in R_PORT_PB_SET is set to enph or if the syncser1 field in R_PORT_PB_SET is set to ss1extra.	0 = port 1 = cs
3	cs3	This bit determines whether pb3 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
2	cs2	This bit determines whether pb2 is a general I/O or peripheral chip-select signal.	0 = port 1 = cs
1	scsi1	This bit determines whether signal $\overline{\text{s1enph}}$ at port SCSI-8 p1 is output on pb7. In SCSI-W mode, the scsi1 field determines whether signal s0enloid is output on pb7. The setting of the scsi1 field overrides that of the cs7, dir7 and syncser3 fields in R_PORT_PB_SET	0 = port_cs 1 = enph
0	scsi0	This bit determines whether signal $\overline{\text{s0enph}}$ at port SCSI-8 p0 is output on pb4. The setting of the scsi0 field overrides that of the cs4, dir4 and syncser1 fields in R_PORT_PB_SET.	0 = port_cs 1 = enph

Note: This is an 8 bit wide register that is part of register R_PORT_PB_SET.

18.7.9 R_PORT_PB_I2C

General Port PB I2C Register, General Characteristics

ID of register	R_PORT_PB_I2C	**Size**	8 bits
Offset	0x3B	**Read/Write**	Write only
Address	0xB000003B	**Initial value**	0

Bit Assignments of R_PORT_PB_I2C

Bit(s)	Name	Description	State/Range
7 - 6	Reserved	-	
5	syncser3	If this bit is set, pb7 is used for synchronous serial port p3, which in some modes needs an extra signal. The setting of the syncser3 field overrides that of the cs7 and dir7 fields in R_PORT_PB_SET. The setting of the syncser3 field is ignored if the scsi1 field in R_PORT_PB_SET is set to enph.	0 = port_cs 1 = ss3extra
4	syncser1	If this bit is set, pb4 is used for synchronous serial port p1, which in some modes needs an extra signal. The setting of the syncser1 field overrides that of the cs4 and dir4 fields in R_PORT_PB_SET. The setting of the syncser1 field is ignored if the scsi0 field in R_PORT_PB_SET is set to enph.	0 = port_cs 1 = ss1extra
3	i2c_en	This bit enables or disables the I2C mode at pb1 and pb0.	0 = off 1 = on
2	i2c_d	This bit is the I2C output data bit. When the I2C mode is selected (field i2c_en = on) and the output is enabled (field i2c_oe_= enable), i2c_d is output on pb0. The i2c_d field is ignored if the I2C mode is not enabled (i2c_en = off). (note 1)	0 - 1
1	i2c_clk	This bit is the I2C clock signal. When the I2C mode is selected (field i2c_en = on), i2c_clk is output on pb1. The i2c_clk field is ignored if the I2C mode is not enabled (i2c_en = off).	0 - 1
0	i2c_oe_	This bit is the output enable for the I2C mode. When the I2C mode is selected (field i2c_en = on), this field enables or disables the output at pb0. The i2c_oe_ field is ignored if the I2C mode is not enabled (i2c_en = off).	0 = enable 1 = disable

Note 1: The I2C data bit should normally be an open collector output. Open collector behavior is achieved by setting the i2c_d bit to 0 and using the i2c_oe_field to control the data output.

Note 2: This is an 8 bit wide register that is part of 32-bit register R_PORT_PB_SET.

18.7.10 R_PORT_PB_READ

General Port PB Read Register, General Characteristics

ID of register	R_PORT_PB_READ	**Size**	32 bits
Offset	0x38	**Read/Write**	Read only
Address	0xB0000038	**Initial value**	Unknown

Bit Assignments of R_PORT_PB_READ

Bit(s)	Name	Description	State/Range
31 - 8	Reserved		
7 - 0	data_in	Data in from general port PB.	0 - 255

18.8 Serial Port Registers

18.8.1 R_SERIAL0_CTRL

Serial Port 0 Control Register, General Characteristics

ID of register	R_SERIAL0_CTRL	**Size**	32 bits
Offset	0x60	**Read/Write**	Write only
Address	0xB0000060	**Initial value**	Bit 14 and 22 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL0_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
27 - 24	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
23	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
22	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable

Bit Assignments of R_SERIAL0_CTRL (continued)

Bit(s)	Name	Description	State/Range
21	rts_	This bit controls the rts pin for the serial port.	0 = active 1 = inactive
20	sampling	This bit determines the sampling mode for the serial receiver.If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority
19	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
18	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
17	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
16	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit
15	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
14	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
13	auto_cts	This bit enables automatic cts handling. If set (active), a high signal on cts stops transmission after the ongoing byte.	0 = disabled 1 = active
12	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
11	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
10	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
9	tr_par_en	This bit enables the parity for the serial transmitter.	0 = disable 1 = enable
8	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit
7 - 0	data_out	Data out to serial port 0.	

Note: For the serial port to operate correctly, register R_SERIAL0_XOFF must also be initialized.

18.8.2 R_SERIAL0_BAUD

Serial Port 0 Baud Register, General Characteristics

ID of register	R_SERIAL0_BAUD	**Size**	8 bits
Offset	0x63	**Read/Write**	Write only
Address	0xB0000063	**Initial value**	Unknown

Bit Assignments of R_SERIAL0_BAUD

Bit(s)	Name	Description	State/Range
7 - 4	tr_baud	This 4-bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
3 - 0	rec_baud	This 4-bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved

Note: This 8 bit wide register is part of register R_SERIAL0_CTRL.

18.8.3 R_SERIAL0_REC_CTRL

Serial Port 0 Receive Control Register, General Characteristics

ID of register	R_SERIAL0_REC_CTRL	**Size**	8 bits
Offset	0x62	**Read/Write**	Write only
Address	0xB0000062	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL0_REC_CTRL

Bit(s)	Name	Description	State/Range
7	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
6	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
5	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
4	sampling	This bit determines the sampling mode for the serial receiver.If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority
3	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
2	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
1	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
0	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit

Note: This 8 bit wide register is part of register R_SERIAL0_CTRL.

18.8.4 R_SERIAL0_TR_CTRL

Serial Port 0 Transmit Control Register, General Characteristics

ID of register	R_SERIAL0_TR_CTRL	**Size**	8 bits
Offset	0x61	**Read/Write**	Write only
Address	0xB0000061	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL0_TR_CTRL

Bit(s)	Name	Description	State/Range
7	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
6	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
5	auto_cts	This bit enables automatic $\overline{cts}$ handling. If set (active), a high signal on $\overline{cts}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
4	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
3	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
2	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
1	tr_par_en	This bit enables/disables the parity for the serial transmitter.	0 = disable 1 = enable
0	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit

Note: This 8 bit wide register is a part of register R_SERIAL0_CTRL.

18.8.5 R_SERIAL0_TR_DATA

Serial Port 0 Transmit Data Register, General Characteristics

ID of register	R_SERIAL0_TR_DATA	**Size**	8 bits
Offset	0x60	**Read/Write**	Write only
Address	0xB0000060	**Initial value**	Unknown

Bit Assignments of R_SERIAL0_TR_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Data out to serial port 0	

Note: This 8 bit wide register is a part of register R_SERIAL0_CTRL.

18.8.6 R_SERIAL0_READ

Serial Port 0 Read Register, General Characteristics

ID of register	R_SERIAL0_READ	**Size**	32 bits
Offset	0x60	**Read/Write**	Read only
Address	0xB0000060	**Initial value**	Unknown

Bit Assignments of R_SERIAL0_READ

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	
15	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL0_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL0_XOFF is set to enable.	0 = no_xoff 1 = xoff
14	cts_	This bit gives the value on the $\overline{cts}$ pin.	0 = active 1 = inactive
13	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
12	rxd	This bit gives the value on the rxd pin.	
11	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
10	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
9	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
8	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in this register and in R_SERIAL0_STATUS.	

18.8.7 R_SERIAL0_STATUS

Serial Port 0 Status Register, General Characteristics

ID of register	R_SERIAL0_STATUS	**Size**	8 bits
Offset	0x61	**Read/Write**	Read only
Address	0xB0000061	**Initial value**	Unknown

Bit Assignments of R_SERIAL0_STATUS

Bit(s)	Name	Description	State/Range
7	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL0_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL0_XOFF is set to enable.	0 = no_xoff 1 = xoff
6	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin	0 = active 1 = inactive
5	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
4	rxd	This bit gives the value on the rxd pin.	
3	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field in R_SERIAL0_REC_DATA or in R_SERIAL0_READ is read.	0 = no 1 = yes
2	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL0_REC_DATA or in R_SERIAL0_READ is read.	0 = no 1 = yes
1	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL0_REC_DATA or in R_SERIAL0_READ is read.	0 = no 1 = yes
0	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field in R_SERIAL0_REC_DATA or in R_SERIAL0_READ is read.	0 = no 1 = yes

Note: This 8 bit wide register is a part of register R_SERIAL0_READ.

18.8.8 R_SERIAL0_REC_DATA

Serial Port 0 Receive Data Register, General Characteristics

ID of register	R_SERIAL0_REC_DATA	**Size**	8 bits
Offset	0x60	**Read/Write**	Read only
Address	0xB0000060	**Initial value**	Unknown

Bit Assignments of R_SERIAL0_REC_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in R_SERIAL0_STATUS and in R_SERIAL0_READ.	

Note: This 8 bit wide register is a part of register R_SERIAL0_READ.

18.8.9 R_SERIAL0_XOFF

Serial Port 0 XOFF Register, General Characteristics

ID of register	R_SERIAL0_XOFF	Size	32 bits
Offset	0x64	Read/Write	Write only
Address	0xB0000064	Initial value	Unknown

Bit Assignments of R_SERIAL0_XOFF

Bit(s)	Name	Description	State/Range
31 - 10	Reserved	-	0
9	tx_stop	When this bit is set, the serial transmitter stops after the ongoing byte.	0 = enable 1 = stop
8	auto_xoff	This bit enables/disables automatic xoff handling.	0 = disable 1 = enable
7 - 0	xoff_char	The code for the xoff character.	

Note: Writing to bits 15:8 of this register will clear the xoff_detect bit in register R_SERIAL0_READ.

18.8.10 R_SERIAL1_CTRL

Serial Port 1 Control Register, General Characteristics

ID of register	R_SERIAL1_CTRL	**Size**	32 bits
Offset	0x68	**Read/Write**	Write only
Address	0xB0000068	**Initial value**	Bit 14 and 22 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL1_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
27 - 24	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
23	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
22	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
21	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
20	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority

Bit Assignments of R_SERIAL1_CTRL (continued)

Bit(s)	Name	Description	State/Range
19	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
18	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
17	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
16	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit
15	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
14	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
13	auto_cts	This bit enables automatic $\overline{cts}$ handling. If set (active), a high signal on $\overline{cts}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
12	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
11	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
10	tr_par	This bits selects the parity for the serial transmitter.	0 = even 1 = odd
8	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit
7 - 0	data_out	Data out to serial port 1.	

Note: For this serial port to operate correctly, register R_SERIAL1_XOFF must also be initialized.

18.8.11 R_SERIAL1_BAUD

Serial Port 1 Baud Register, General Characteristics

ID of register	R_SERIAL1_BAUD	**Size**	8 bits
Offset	0x6B	**Read/Write**	Write only
Address	0xB000006B	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_BAUD

Bit(s)	Name	Description	State/Range
7 - 4	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
3 - 0	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved

Note: This 8 bit wide register is part of register R_SERIAL1_CTRL.

18.8.12 R_SERIAL1_REC_CTRL

Serial Port 1 Receive Control Register, General Characteristics

ID of register	R_SERIAL1_REC_CTRL	**Size**	8 bits
Offset	0x6A	**Read/Write**	Write only
Address	0xB000006A	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL1_REC_CTRL

Bit(s)	Name	Description	State/Range
7	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
6	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
5	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
4	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority
3	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
2	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
1	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
0	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit

Note: This 8 bit wide register is part of register R_SERIAL1_CTRL.

18.8.13 R_SERIAL1_TR_CTRL

Serial Port 1 Transmit Control Register, General Characteristics

ID of register	R_SERIAL1_TR_CTRL	**Size**	8 bits
Offset	0x69	**Read/Write**	Write only
Address	0xB0000069	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL1_TR_CTRL

Bit(s)	Name	Description	State/Range
7	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
6	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
5	auto_cts	This bit enables automatic $\overline{\text{cts}}$ handling. If set (active), a high signal on $\overline{\text{cts}}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
4	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
3	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
2	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
1	tr_par_en	This bit enables/disables the parity for the serial transmitter.	0 = disable 1 = enable
0	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit

Note: This 8 bit wide register is part of register R_SERIAL1_CTRL.

18.8.14 R_SERIAL1_TR_DATA

Serial Port 1 Transmit Data Register, General Characteristics

ID of register	R_SERIAL1_TR_DATA	**Size**	8 bits
Offset	0x68	**Read/Write**	Write only
Address	0xB0000068	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_TR_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Data out to serial port 1.	

Note: This 8 bit wide register is part of register R_SERIAL1_CTRL.

18.8.15 R_SERIAL1_READ

Serial Port 1 Read Register, General Characteristics

ID of register	R_SERIAL1_READ	**Size**	32 bits
Offset	0x68	**Read/Write**	Read only
Address	0xB0000068	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_READ

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	
15	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL1_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL1_XOFF is set to enable.	0 = no_xoff 1 = xoff
14	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin.	0 = active 1 = inactive
13	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
12	rxd	This bit gives the value on the rxd pin.	
11	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
10	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
9	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
8	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in this register and in R_SERIAL1_STATUS.	

18.8.16 R_SERIAL1_STATUS

Serial Port 1 Status Register, General Characteristics

ID of register	R_SERIAL1_STATUS	**Size**	8 bits
Offset	0x69	**Read/Write**	Read only
Address	0xB0000069	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_STATUS

Bit(s)	Name	Description	State/Range
7	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL1_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL1_XOFF is set to enable.	0 = no_xoff 1 = xoff
6	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin.	0 = active 1 = inactive
5	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
4	rxd	This bit gives the value on the rxd pin.	
3	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field in R_SERIAL1_REC_DATA or in R_SERIAL1_READ is read.	0 = no 1 = yes
2	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL1_REC_DATA or in R_SERIAL1_READ is read.	0 = no 1 = yes
1	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL1_REC_DATA or in R_SERIAL1_READ is read.	0 = no 1 = yes
0	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field in R_SERIAL1_REC_DATA or in R_SERIAL1_READ is read.	0 = no 1 = yes

Note: This 8 bit wide register a part of register R_SERIAL1_READ.

18.8.17 R_SERIAL1_REC_DATA

Serial Port I Receive Data Register, General Characteristics

ID of register	R_SERIAL1_REC_DATA	**Size**	8 bits
Offset	0x68	**Read/Write**	Read only
Address	0xB0000068	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_REC_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in R_SERIAL1_STATUS and in R_SERIAL1_READ.	

Note: This 8 bit wide register is part of register R_SERIAL1_READ.

18.8.18 R_SERIAL1_XOFF

Serial Port 1 XOFF Register, General Characteristics

ID of register	R_SERIAL1_XOFF	**Size**	32 bits
Offset	0x6C	**Read/Write**	Write only
Address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SERIAL1_XOFF

Bit(s)	Name	Description	State/Range
31 - 10	Reserved	-	0
9	tx_stop	When this bit is set, the serial transmitter stops after the ongoing byte.	0 = enable 1 = stop
8	auto_xoff	This bit enables/disables automatic xoff handling.	0 = disable 1 = enable
7 - 0	xoff_char	The code for the xoff character.	

Note: Writing to bits 15:8 of this register will clear the xoff_detect bit in register R_SERIAL1_READ.

18.8.19 R_SERIAL2_CTRL

Serial Port 2 Control Register, General Characteristics

ID of register	R_SERIAL2_CTRL	**Size**	32 bits
Offset	0x70	**Read/Write**	Write only
Address	0xB0000070	**Initial value**	Bit 14 and 22 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL2_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	This 4-bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
27 - 24	rec_baud	This 4-bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
23	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
22	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
21	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
20	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority

Bit Assignments of R_SERIAL2_CTRL (continued)

Bit(s)	Name	Description	State/Range
19	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
18	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
17	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
16	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit
15	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
14	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
13	auto_cts	This bit enables automatic $\overline{\text{cts}}$ handling. If set (active), a high signal on $\overline{\text{cts}}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
12	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
11	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
10	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
9	tr_par_en	This bit enables/disables the parity for the serial transmitter.	0 = disable 1 = enable
8	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit
7 - 0	data_out	Data out to serial port 2.	

Note: For this serial port to operate correctly, register R_SERIAL2_XOFF must also be initialized.

18.8.20 R_SERIAL2_BAUD

Serial Port 2 Baud Register, General Characteristics

ID of register	R_SERIAL2_BAUD	**Size**	8 bits
Offset	0x73	**Read/Write**	Write only
Address	0xB0000073	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_BAUD

Bit(s)	Name	Description	State/Range
7 - 4	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
3 - 0	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 8 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved

Note: This 8 bit wide register is part of register R_SERIAL2_CTRL.

18.8.21 R_SERIAL2_REC_CTRL

Serial Port 2 Receive Control Register, General Characteristics

ID of register	R_SERIAL2_REC_CTRL	Size	8 bits
Offset	0x72	Read/Write	Write only
Address	0xB0000072	Initial value	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL2_REC_CTRL

Bit(s)	Name	Description	State/Range
7	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
6	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
5	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
4	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority
3	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
2	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
1	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
0	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit

Note: This 8 bit wide register is part of register R_SERIAL2_CTRL.

18.8.22 R_SERIAL2_TR_CTRL

Serial Port 2 Transmit Control Register, General Characteristics

ID of register	R_SERIAL2_TR_CTRL	**Size**	8 bits
Offset	0x71	**Read/Write**	Write only
Address	0xB0000071	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL2_TR_CTRL

Bit(s)	Name	Description	State/Range
7	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
6	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
5	auto_cts	This bit enables automatic $\overline{\text{cts}}$ handling. If set (active), a high signal on $\overline{\text{cts}}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
4	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
3	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
2	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
1	tr_par_en	This bit enables/disables parity for the serial transmitter.	0 = disable 1 = enable
0	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit

Note: This 8 bit wide register is part of register R_SERIAL2_CTRL.

18.8.23 R_SERIAL2_TR_DATA

Serial Port 2 Transmit Data Register, General Characteristics

ID of register	R_SERIAL2_TR_DATA	**Size**	8 bits
Offset	0x70	**Read/Write**	Write only
Address	0xB0000070	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_TR_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Data out to serial port 2.	

Note: This 8 bit wide register is part of register R_SERIAL2_CTRL.

18.8.24 R_SERIAL2_READ

Serial Port 2 Read Register, General Characteristics

ID of register	R_SERIAL2_READ	**Size**	32 bits
Offset	0x70	**Read/Write**	Read only
Address	0xB0000070	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_READ

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	
15	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL2_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL2_XOFF is set to enable.	0 = no_xoff 1 = xoff
14	cts_	This bit gives the value on the $\overline{cts}$ pin.	0 = active 1 = inactive
13	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
12	rxd	This bit gives the value on the rxd pin.	
11	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
10	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
9	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
8	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in this register and in R_SERIAL2_STATUS.	

18.8.25 R_SERIAL2_STATUS

Serial Port 2 Status Register, General Characteristics

ID of register	R_SERIAL2_STATUS	**Size**	8 bits
Offset	0x71	**Read/Write**	Read only
Address	0xB0000071	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_STATUS

Bit(s)	Name	Description	State/Range
7	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL2_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL2_XOFF is set to enable.	0 = no_xoff 1 = xoff
6	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin.	0 = active 1 = inactive
5	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
4	rxd	This bit gives the value on the rxd pin.	
3	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field in R_SERIAL2_REC_DATA or in R_SERIAL2_READ is read.	0 = no 1 = yes
2	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL2_REC_DATA or in R_SERIAL2_READ is read.	0 = no 1 = yes
1	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL2_REC_DATA or in R_SERIAL2_READ is read.	0 = no 1 = yes
0	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field in R_SERIAL2_REC_DATA or in R_SERIAL2_READ is read.	0 = no 1 = yes

Note: This 8 bit wide register is part of register R_SERIAL2_READ.

18.8.26 R_SERIAL2_REC_DATA

Serial Port 2 Receive Data Register, General Characteristics

ID of register	R_SERIAL2_REC_DATA	**Size**	8 bits
Offset	0x70	**Read/Write**	Read only
Address	0xB0000070	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_REC_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in R_SERIAL2_STATUS and in R_SERIAL2_READ.	

Note: This 8 bit wide register is part of register R_SERIAL2_READ.

18.8.27 R_SERIAL2_XOFF

Serial Port 2 XOFF Register, General Characteristics

ID of register	R_SERIAL2_XOFF	**Size**	32 bits
Offset	0x116	**Read/Write**	Write only
Address	0xB0000116	**Initial value**	Unknown

Bit Assignments of R_SERIAL2_XOFF

Bit(s)	Name	Description	State/Range
31 - 10	Reserved	-	0
9	tx_stop	When this bit is set, the serial transmitter stops after the ongoing byte.	0 = enable 1 = stop
8	auto_xoff	This bit enables/disables automatic xoff handling.	0 = disable 1 = enable
7 - 0	xoff_char	The code for the xoff character.	

Note: Writing to bits 15:8 of this register will clear the xoff_detect bit in register R_SERIAL2_READ.

18.8.28 R_SERIAL3_CTRL

Serial Port 3 Control Register, General Characteristics

ID of register	R_SERIAL3_CTRL	**Size**	32 bits
Offset	0x78	**Read/Write**	Write only
Address	0xB0000078	**Initial value**	Bit 14 and 22 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL3_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
27 - 24	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
23	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
22	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
21	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
20	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority

Bit Assignments of R_SERIAL3_CTRL (continued)

Bit(s)	Name	Description	State/Range
19	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ.	0 = normal 1 = stick
18	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
17	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
16	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit
15	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
14	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
13	auto_cts	This bit enables automatic $\overline{\text{cts}}$ handling. If set (active), a high signal on $\overline{\text{cts}}$ stops transmission after the ongoing byte.	0 = disabled 1 = active
12	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
11	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
10	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
9	tr_par_en	This bit enables/disables the parity for the serial transmitter.	0 = disable 1 = enable
8	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit
7 - 0	data_out	Data out to serial port 3.	

Note: For this serial port to operate correctly, register R_SERIAL3_XOFF must also be initialized.

18.8.29 R_SERIAL3_BAUD

Serial Port 3 Baud Register, General Characteristics

ID of register	R_SERIAL3_BAUD	**Size**	8 bits
Offset	0x7B	**Read/Write**	Write only
Address	0xB000007B	**Initial value**	Unknown

Bit Assignments of R_SERIAL3_BAUD

Bit(s)	Name	Description	State/Range
7 - 4	tr_baud	This 4 bit field is used to select the baud rate for the transmitter.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved
3 - 0	rec_baud	This 4 bit field is used to select the baud rate for the receiver.	0 = c300Hz 1 = c600Hz 2 = c1200Hz 3 = c2400Hz 4 = c4800Hz 5 = c9600Hz 6 = c19k2Hz 7 = c38k4Hz 8 = c57k6Hz 9 = c115k2Hz 10 = c230k4Hz 11 = c460k8Hz 12 = c921k6Hz 13 = c1843k2Hz 14 = c6250kHz 15 = reserved

Note: This 8 bit wide register is part of register R_SERIAL3_CTRL.

18.8.30 R_SERIAL3_REC_CTRL

Serial Port 3 Receive Control Register, General Characteristics

ID of register	R_SERIAL3_REC_CTRL	**Size**	8 bits
Offset	0x7A	**Read/Write**	Write only
Address	0xB000007A	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL3_REC_CTRL

Bit(s)	**Name**	**Description**	**State/Range**
7	dma_err	This bit controls the handling of receive errors when DMA is used. If this bit is set to stop, a receive error generates an end_of_packet to the DMA. The erroneous byte is not entered into the fifo. If the bit is set to ignore, receive errors are ignored when DMA is used.	0 = stop 1 = ignore
6	rec_enable	This bit enables/disables the serial receiver.	0 = disable 1 = enable
5	rts_	This bit controls the $\overline{\text{rts}}$ pin for the serial port.	0 = active 1 = inactive
4	sampling	This bit determines the sampling mode for the serial receiver. If set to middle, one sample is taken in the middle of each data bit. If set to majority, the majority of three samples in the middle of each data bit is taken.	0 = middle 1 = majority
3	rec_stick_par	This bit selects normal or stick parity for the serial receiver. If set to normal, normal parity checking is used. If set to stick, parity is checked as a logic 0 or 1, compared to the value of rec_par. A parity error is generated if the values differ	0 = normal 1 = stick
2	rec_par	This bit selects the parity for the serial receiver.	0 = even 1 = odd
1	rec_par_en	This bit enables/disables the parity for the serial receiver.	0 = disable 1 = enable
0	rec_bitnr	This bit determines the number of data bits for the serial receiver.	0 = rec_8bit 1 = rec_7bit

Note: This 8 bit wide register is part of register R_SERIAL3_CTRL.

18.8.31 R_SERIAL3_TR_CTRL

Serial Port 3 Transmit Control Register, General Characteristics

ID of register	R_SERIAL3_TR_CTRL	**Size**	8 bits
Offset	0x79	**Read/Write**	Write only
Address	0xB0000079	**Initial value**	Bit 6 set to 0. Other bits unknown.

Bit Assignments of R_SERIAL3_TR_CTRL

Bit(s)	Name	Description	State/Range
7	txd	This bit determines the value of the txd pin when the serial transmitter is disabled.	
6	tr_enable	This bit enables/disables the serial transmitter.	0 = disable 1 = enable
5	auto_cts	This bit enables automatic $\overline{\text{cts}}$ handling. If set (active), a high signal on $\overline{\text{cts}}$ stops transmission after the ongoing byte	0 = disabled 1 = active
4	stop_bits	This bit determines the number of stop bits for the serial transmitter.	0 = one_bit 1 = two_bits
3	tr_stick_par	This bit selects normal or stick parity mode for the serial transmitter. If set to normal, normal parity generation is used. If set to stick, parity is set to a logic 0 or 1, depending on the value of tr_par.	0 = normal 1 = stick
2	tr_par	This bit selects the parity for the serial transmitter.	0 = even 1 = odd
0	tr_bitnr	This bit determines the number of data bits for the serial transmitter.	0 = tr_8bit 1 = tr_7bit

Note: This 8 bit wide register is part of register R_SERIAL3_CTRL.

18.8.32 R_SERIAL3_TR_DATA

Serial Port 3 Transmit Data Register, General Characteristics

ID of register	R_SERIAL3_TR_DATA	Size	8 bits
Offset	0x78	Read/Write	Write only
Address	0xB0000078	Initial value	Unknown

Bit Assignments of R_SERIAL3_TR_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_out	Data out to serial port 3.	

Note: This 8 bit wide register is part of register R_SERIAL3_CTRL.

18.8.33 R_SERIAL3_READ

Serial Port 3 Read Register, General Characteristics

ID of register	R_SERIAL3_READ	**Size**	32 bits
Offset	0x78	**Read/Write**	Read only
Address	0xB0000078	**Initial value**	Unknown

Bit Assignments of R_SERIAL3_READ

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	
15	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL3_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL3_XOFF is set to enable.	0 = no_xoff 1 = xoff
14	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin.	0 = active 1 = inactive
13	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
12	rxd	This bit gives the value on the rxd pin.	
11	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
10	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
9	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
8	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field of this register is read.	0 = no 1 = yes
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in this register and in R_SERIAL3_STATUS.	

18.8.34 R_SERIAL3_STATUS

General Characteristics of R_SERIAL3_STATUS

ID of register	R_SERIAL3_STATUS	**Size**	8 bits
Offset	0x79	**Read/Write**	Read only
Address	0xB0000079	**Initial value**	Unknown

Bit Assignments of R_SERIAL3_STATUS

Bit(s)	Name	Description	State/Range
7	xoff_detect	This bit is set if the xoff character is detected in the received data. The bit is cleared by writing to bits 15:8 of R_SERIAL3_XOFF. The data written doesn't matter for the clear operation. xoff will only be detected if bit auto_xoff in R_SERIAL3_XOFF is set to enable.	0 = no_xoff 1 = xoff
6	cts_	This bit gives the value on the $\overline{\text{cts}}$ pin.	0 = active 1 = inactive
5	tr_ready	If this bit is set (ready), the serial transmitter is ready and one byte can be written to it.	0 = full 1 = ready
4	rxd	This bit gives the value on the rxd pin.	
3	overrun	This bit is set when an overrun error is detected in the serial receiver. The bit is cleared when the data_in field in R_SERIAL3_REC_DATA or in R_SERIAL3_READ is read.	0 = no 1 = yes
2	par_err	This bit is set when a parity error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL3_REC_DATAor in R_SERIAL3_READ is read.	0 = no 1 = yes
1	framing_err	This bit is set when a framing error is detected by the serial receiver. The bit is cleared when the data_in field in R_SERIAL3_REC_DATAor in R_SERIAL3_READ is read.	0 = no 1 = yes
0	data_avail	This bit is set when data is available from the serial receiver. The bit is cleared when the data_in field in R_SERIAL3_REC_DATAor in R_SERIAL3_READ is read.	0 = no 1 = yes

Note: This 8 bit wide register is part of register R_SERIAL3_READ.

18.8.35 R_SERIAL3_REC_DATA

Serial Port 3 Receive Data Register, General Characteristics

ID of register	R_SERIAL3_REC_DATA	**Size**	8 bits
Offset	0x78	**Read/Write**	Read only
Address	0xB0000078	**Initial value**	Unknown

Bit Assignments of R_SERIAL3_REC_DATA

Bit(s)	Name	Description	State/Range
7 - 0	data_in	Data in from the serial receiver. Reading this register will clear the data_avail, overrun, par_err and framing_error bits in R_SERIAL3_STATUS and in R_SERIAL3_READ.	

Note: This 8 bit wide register is part of register R_SERIAL3_READ.

18.8.36 R_SERIAL3_XOFF

Serial Port 3 XOFF Register, General Characteristics

ID of register	R_SERIAL3_XOFF	**Size**	32 bits
Offset	0x7C	**Read/Write**	Write only
Address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SERIAL3_XOFF

Bit(s)	Name	Description	State/Range
31 - 10	Reserved	-	0
9	tx_stop	When this bit is set, the serial transmitter stops after the ongoing byte.	0 = enable 1 = stop
8	auto_xoff	This bit enables/disables automatic xoff handling.	0 = disable 1 = enable
7 - 0	xoff_char	The code for the xoff character.	

Note: Writing to bit 15:8 of this register will clear the xoff_detect bit in register R_SERIAL3_READ.

18.8.37 R_ALT_SER_BAUDRATE

Alternative Serial Baud Rate Register, General Characteristics

ID of register	R_ALT_SER_BAUDRATE	**Size**	32 bits
Offset	0x5C	**Read/Write**	Write only
Address	0xB000005C	**Initial value**	0

Bit Assignments of R_ALT_SER_BAUDRATE

Bit(s)	Name	Description	State/Range
31 - 30	Reserved	-	0
29 - 28	ser3_tr	Alternative baud rates for the serial port 3 transmitter can be chosen by setting the value of this field. If set to normal (default) the baud rate is chosen by the R_SERIAL3_BAUD register. prescale overrides R_SERIAL3_BAUD and uses the baud rate generated by R_SERIAL_PRESCALE (user baud rate). extern overrides R_SERIAL3_BAUD and uses the external baud rate. timer overrides R_SERIAL3_BAUD and uses the timer0 as baud rate generator.	0 = normal 1 = prescale 2 = extern 3 = timer
27 - 26	Reserved	-	0
25 - 24	ser3_rec	Alternative baud rates for the serial port 3 receiver can be chosen by setting the value of this field. If set to normal (default) the baud rate is chosen by the R_SERIAL3_BAUD register. prescale overrides R_SERIAL3_BAUD and uses the baud rate generated by R_SERIAL_PRESCALE (user baud rate). extern overrides R_SERIAL3_BAUD and uses the external baud rate. timer overrides R_SERIAL3_BAUD and uses the timer0 as baud rate generator.	0 = normal 1 = prescale 2 = extern 3 = timer
23 - 22	Reserved	-	0
21 - 20	ser2_tr	Alternative baud rates for the serial port 2 transmitter can be chosen by setting the value of this field. See ser3_tr.	0 = normal 1 = prescale 2 = extern 3 = timer
19 - 18	Reserved	-	0
17 - 16	ser2_rec	Alternative baud rates for the serial port 2 receiver can be chosen by setting the value of this field. See ser3_rec.	0 = normal 1 = prescale 2 = extern 3 = timer
15 - 14	Reserved	-	0
13 - 12	ser1_tr	Alternative baud rates for the serial port 1 transmitter can be chosen by setting the value of this field. See ser3_tr.	0 = normal 1 = prescale 2 = extern 3 = timer
11 - 10	Reserved	-	0
9 - 8	ser1_rec	Alternative baud rates for the serial port 1 receiver can be chosen by setting the value of this field. See ser3_rec.	0 = normal 1 = prescale 2 = extern 3 = timer
7 - 6	Reserved	-	0

Bit Assignments of R_ALT_SER_BAUDRATE (continued)

Bit(s)	Name	Description	State/Range
5 - 4	ser0_tr	Alternative baud rates for the serial port 0 transmitter can be chosen by setting the value of this field. See ser3_tr.	0 = normal 1 = prescale 2 = extern 3 = timer
3 - 2	Reserved	-	0
1 - 0	ser0_rec	Alternative baud rates for the serial port 0 receiver can be chosen by setting the value of this field. See ser3_rec.	0 = normal 1 = prescale 2 = extern 3 = timer

18.8.38 R_SERIAL_PRESCALE

Serial Prescale Register, General Characteristics

ID of register	R_SERIAL_PRESCALE	**Size**	16 bits
Offset	0xF2	**Read/Write**	Write only
Address	0xB00000F2	**Initial value**	Unknown

Bit Assignments of R_SERIAL_PRESCALE

Bit(s)	Name	Description	State/Range
15 - 0	ser_presc	This fields gives the divide factor for serial clock prescaling. It is used when another baud rate than those predefined in the serial port control registers is needed for the asynchronous serial ports. The usable range is 2 - 65536. A divide factor of 65536 is achieved by setting the field to 0. The resulting baud rate equals 3.125MHz divided by the divide factor. The prescaling starts when the field is written.	0, 2-65535

Note: This is a 16 bit wide register that is part of register R_CLOCK_PRESCALE, see section 18.4 *Timer Registers*.

18.8.39 R_SER_PRESC_STATUS

Serial Prescale Status Register, General Characteristics

ID of register	R_SER_PRESC_STATUS	**Size**	16 bits
Offset	0xF2	**Read/Write**	Read only
Address	0xB00000F2	**Initial value**	Unknown

Bit Assignments of R_SER_PRESC_STATUS

Bit(s)	Name	Description	State/Range
15 - 0	ser_status	Contains the current count value of the serial divide factor.	0-65535

Note: *Table 18-1 This is a 16 bit wide register that is part of register R_PRESCALE_STATUS, see section 18.4 Timer Registers.*

18.9 Network Interface Registers

Mode registers R_NETWORK_SA_0 to R_NETWORK_SA_2 contain two 48-bit station addresses: MA0 and MA1. Each station address can be set to match an individual address, or it can be set to match a single group/multicast address by setting the multicast bit in the address.

The bit ordering convention used here is that address bit 0 (addr[0]) is the first bit transmitted/received on the network and address bit 47 (addr[47]) is the last bit. The multicast bit in Ethernet is addr[0].

The normal way of printing and displaying Ethernet addresses is:
addr[7:0]:addr[15:8]:addr[23:16]:addr[31:24]:addr[39:32]:addr[47:40]
where each byte is printed in hexadecimal.

For more information about the ETRAX 100LX interface, see chapter *9 Network Interface*.

Note: With the IEEE802.5 frame format, any of the addresses MA0 or MA1 are interpreted as a functional address if it matches the functional address format of IEEE802.5.

18.9.1 R_NETWORK_SA_0

Network Station Address Register Part 0, General Characteristics

ID of register	R_NETWORK_SA_0	**Size**	32 bits
Offset	0x80	**Read/Write**	Write only
Register address	0xB0000080	**Initial value**	Unknown

Bit Assignments of R_NETWORK_SA_0

Bit(s)	Name	Description	State/Range
31 - 0	ma0_low	Bit addr[31:0] of station address MA0.	

18.9.2 R_NETWORK_SA_1

Network Station Address Register Part 1, General Characteristics

ID of register	R_NETWORK_SA_1	**Size**	32 bits
Offset	0x84	**Read/Write**	Write only
Register address	0xB0000084	**Initial value**	Unknown

Bit Assignments of R_NETWORK_SA_1

Bit(s)	Name	Description	State/Range
31 - 16	ma1_low	Bit addr[15:0] of station address MA1.	65535
15 - 0	ma0_high	Bit addr[47:32] of station address MA0.	65535

18.9.3 R_NETWORK_SA_2

Network Station Address Register Part 2, General Characteristics

ID of register	R_NETWORK_SA_2	**Size**	32 bits
Offset	0x88	**Read/Write**	Write only
Register address	0xB0000088	**Initial value**	Unknown

Bit Assignments of R_NETWORK_SA_2

Bit(s)	Name	Description	State/Range
31 - 0	ma1_high	Bit addr[47:16] of station address MA1.	

18.9.4 R_NETWORK_GA_0

Network Group Address Table Register Part 0, General Characteristics

ID of register	R_NETWORK_GA_0	**Size**	32 bits
Offset	0x8C	**Read/Write**	Write only
Register address	0xB000008C	**Initial value**	Unknown

Bit Assignments of R_NETWORK_GA

Bit(s)	Name	Description	State/Range
31 - 0	ga_low	Bit [31:0] of the group address table.	

18.9.5 R_NETWORK_GA_1

Network Group Address Table Register Part 1, General Characteristics

ID of register	R_NETWORK_GA_1	**Size**	32 bits
Offset	0x90	**Read/Write**	Write only
Register address	0xB0000090	**Initial value**	Unknown

Bit Assignments of R_NETWORK_GA_1

Bit(s)	Name	Description	State/Range
31 - 0	ga_high	Bit [63:32] of the group address table.	

18.9.6 R_NETWORK_REC_CONFIG

Network Receive Configuration Register, General Characteristics

ID of register	R_NETWORK_REC_CONFIG	**Size**	32 bits
Offset	0x94	**Read/Write**	Write only
Register address	0xB0000094	**Initial value**	Unknown

Bit Assignments of R_NETWORK_REC_CONFIG

Bit(s)	Name	Description	State/Range
31 - 11	Reserved	-	0
10	max_size	This bit determines the maximum packet size for Ethernet packets. If the IEEE 802.1q standard is used, maximum packet size 1522 bytes is selected (size1522). If the IEEE 802.3 standard is used, maximum packet size 1518 bytes is selected (size1518).	0 = size1518 1 = size1522
9	duplex	This bit decides whether full or half duplex mode is used. In full duplex mode, the receiver is not turned off during transmission. The transmitter ignores the col and crs signals. Full duplex flow control is enabled. This mode can also be used for external loopback test.	0 = half 1 = full
8	bad_crc	This bit determines whether to receive or discard frames with CRC errors or alignment errors.	0 = discard 1 = receive
7	oversize	This bit determines whether to receive or discard oversized frames.	0 = discard 1 = receive
6	undersize	This bit determines whether to receive or discard undersized frames.	0 = discard 1 = receive
5	all_roots	This bit determines whether to receive or discard all roots in functional addresses. If set to receive (1), bit [45:32] in functional addresses will be ignored. This bit is only used with IEEE802.5 frame format.	0 = discard 1 = receive
4	tr_broadcast	This bit determines whether to receive or discard Token Ring broadcast frames (address 0xFFFFFFFF000C).	0 = discard 1 = receive
3	broadcast	This bit determines whether to receive or discard broadcast frames (address 0xFFFFFFFFFFFF).	0 = discard 1 = receive
2	individual	If this field is set to receive (1), individual addresses will also be matched with the group address table.	0 = discard 1 = receive
1	ma1	This bit enables or disables station address MA1.	0 = disable 1 = enable
0	ma0	This bit enables or disables station address MA0.	0 = disable 1 = enable

18.9.7 R_NETWORK_GEN_CONFIG

Network General Configuration Register, General Characteristics

ID of register	R_NETWORK_GEN_CONFIG	**Size**	32 bits
Offset	0x98	**Read/Write**	Write only
Register address	0xB0000098	**Initial value**	Unknown

Bit Assignments of R_NETWORK_GEN_CONFIG

Bit(s)	Name	Description	State/Range
31 - 6	Reserved	-	0
5	loopback	This field turns internal loopback mode on or off. If on, the transmitted frames are directly passed to the receiver, and the MII or SNI interface is disabled.The col and crs signals are therefore also disabled in loopback mode, which has the side effect that each transmitted frame will cause the carrier_loss counter in "R_PHY_COUNTERS" to increment. The col signal can however be enabled, for testing purposes, via R_TEST_MODE. The transmit speed in internal loopback mode is 100 Mbit/s. Loop back mode also forces the reception of transmitted frames, regardless of the value of the duplex bit in the "R_NETWORK_REC_CONFIG" register.	0 = off 1 = on
4	frame	If this field is set to ether, frame format according to IEEE802.3 (Ethernet) will be used, if set to tokenr, frame format according to IEEE802.5 (Token Ring) will be used.	0 = ether 1 = tokenr
3	vg	This bit turns IEEE802.12 mode (VG-anylan) mode on (1) or off (0).	0 = off 1 = on
2 - 1	phy	This field determines the physical connection. If set to sni (0), SNI mode is used. All other states give MII mode, but the txer pin will be used in different ways according to the setting of this field. If the field is set to mii_clk, a 25 MHz clock will be output on the txer pin. If the field is set to mii_err, the txer pin will be used to indicate a transmit error. If the field is set to mii_req, the output indicates that an address is recognized.	0 = sni 1 = mii_clk 2 = mii_err 3 = mii_req
0	enable	This bit enables or disables the network controller.	0 = off 1 = on

18.9.8 R_NETWORK_TR_CTRL

Network Transmit Control Register, General Characteristics

ID of register	R_NETWORK_TR_CTRL	**Size**	32 bits
Offset	0x9C	**Read/Write**	Write only
Register address	0xB000009C	**Initial value**	Unknown

Bit Assignments of R_NETWORK_TR_CTRL

Bit(s)	Name	Description	State/Range
31 - 9	Reserved	-	0
8	clr_error	This bit clears underrun and excessive collision conditions. If the transmitter has stopped due to either error condition it is restarted by setting this bit to clr. The bit is not saved, i.e. the state reverts to nop. There is no need to set it to nop again to be able to detect a new underrun.	0 = nop 1 = clr
7 - 6	Reserved	-	0
5	delay	In IEEE802.12 mode this bit enables a 2 µs delay from transmit request acknowledge until transmission starts. (note 1)	0 = none 1 = d2us
4	cancel	This bit cancels a pending frame. If it is set to do (1), the transmitter completes the current transmission attempt (if any), and then stops. The excessive retry condition is then entered, regardless of whether a transmission was in progress.	0 = dont 1 = do
3	cd	In IEEE802.12 mode this bit determines if a transmit request acknowledge will occur on the collision detect (ack_col) pin or the carrier sense (ack_crs) pin. (note 1)	0 = ack_col 1 = ack_crs
2	retry	This bit enables or disables retransmission. If it is set, the transmitter will stop and enter the excessive retry condition after the first collision instead of making 15 transmission retries. This bit is ignored in IEEE802.12 mode.	0 = enable 1 = disable
1	pad	If this bit is set to enable, a frame shorter than 60 bytes (excluding preamble, start of frame delimiter and CRC) is padded to 60 bytes. The pad consists of all 0's. This bit should be 0 in IEEE802.12 mode.	0 = disable 1 = enable
0	crc	This bit enables or disables CRC. If it is set, the transmitter will not add the CRC after the frame.	0 = enable 1 = disable

Note: In IEEE802.3 (Ethernet) mode this bit is reserved and should be set to 0.

18.9.9 R_NETWORK_MGM_CTRL

Network MGM Control Register, General Characteristics

ID of register	R_NETWORK_MGM_CTRL	**Size**	32 bits
Offset	0xA0	**Read/Write**	Write only
Register address	0xB00000A0	**Initial value**	Unknown

Bit Assignments of R_NETWORK_MGM_CTRL

Bit(s)	Name	Description	State/Range
31 - 8	Reserved	-	0
7 - 4	txd_pins	These bits are output on txdata3 - txdata0 when the network controller is disabled or in internal loopback mode. Bits 7 - 5 are output on txdata3 - txdata1 also when SNI mode is selected.	0 - 15
3	txer_pin	This bit is output on the txer pin when SNI mode is selected.	0 - 1
2	mdck	Management clock. This bit is output on the mdc pin.	0 - 1
1	mdoe	This bit is the output enable for the mdio pin.	0=disable 1=enable
0	mdio	mdio pin output data.	0 - 1

18.9.10 R_NETWORK_STAT

Network Status Register, General Characteristics

ID of register	R_NETWORK_STAT	**Size**	32 bits
Offset	0xA0	**Read/Write**	Read only
Register address	0xB00000A0	**Initial value**	Unknown

Bit Assignments of R_NETWORK_STAT

Bit(s)	Name	Description	State/Range
31 - 8	Reserved	-	0
7 - 4	rxd_pins	This field shows the values on rxdata3 - rxdata0.	0 - 15
3	rxer	This field shows the value on the rxer pin.	0 - 1
2	underrun	This bit is set if a transmitter underrun is detected, i.e. it is set if DMA channel 0 is not capable to keep up with the speed of the transmitter. It is cleared by setting the clr_error bit in the “R_NETWORK_TR_CTRL” register.	0 = no 1 = yes
1	exc_col	This bit indicates when excessive collision has been detected. It is set after 15 unsuccessful transmission retries, or after the first collision if retransmission is disabled. It is also set when the transmitter stops after the transmit cancel bit has been set. It is cleared by setting the clr_error bit in the “R_NETWORK_TR_CTRL” register.	0 = no 1 = yes
0	mdio	mdio pin input data.	0 - 1

18.9.11 R_REC_COUNTERS

Receive Error Counters Register, General Characteristics

ID of register	R_REC_COUNTERS	**Size**	32 bits
Offset	0xA4	**Read/Write**	Read only
Register address	0xB00000A4	**Initial value**	Unknown

Bit Assignments of R_REC_COUNTERS

Bit(s)	Name	Description	State/Range
31 - 24	congestion	This field gives the number of otherwise correct frames that were not received due to a FIFO full condition.	0 - 255
23 - 16	oversize	This field gives the number of oversized frames.	0 - 255
15 - 8	alignment_error	This field gives the number of frames with alignment errors.	0 - 255
7 - 0	crc_error	This field gives the number of frames with CRC errors.	0 - 255

Note: Each 8-bit counter is reset by reading it.

18.9.12 R_TR_COUNTERS

Transmit Statistics Counters Register, General Characteristics

ID of register	R_TR_COUNTERS	**Size**	32 bits
Offset	0xA8	**Read/Write**	Read only
Register address	0xB00000A8	**Initial value**	Unknown

Bit Assignments of R_TR_COUNTERS

Bit(s)	Name	Description	State/Range
31 - 24	deferred	This field gives the number of deferred transmit frames.	0 - 255
23 - 16	late_col	This field gives the number of frames that were involved in late collisions.	0 - 255
15 - 8	multiple_col	This field gives the number of frames that were involved in more than one collision.	0 - 255
7 - 0	single_col	This field gives the number of frames that were involved in exactly one collision.	0 - 255

Note: Each 8-bit counter is reset by reading it.

18.9.13 R_PHY_COUNTERS

PHY Error Counters Register, General Characteristics

ID of register	R_PHY_COUNTERS	**Size**	32 bits
Offset	0xAC	**Read/Write**	Read only
Register address	0xB00000AC	**Initial value**	Unknown

Bit Assignments of R_PHY_COUNTERS

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	0
15 - 8	sqe_test_error	When the chip is configured for10Mb Ethernet, this field gives the number of transmit frames for which the sqe test signal was not recognized.	0 - 255
7 - 0	carrier_loss	This field gives the number of transmit frames for which the carrier sense signal was not constantly present during the transmission.	0 - 255

Note: Each 8-bit counter is reset by reading it.

18.10 Parallel Port Registers

18.10.1 R_PAR0_CTRL_DATA

Parallel Port p0 Control and Data Register, General Characteristics

ID of register	R_PAR0_CTRL_DATA	**Size**	32 bits
Offset	0x40	**Read/Write**	Write only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_PAR0_CTRL_DATA

Bit(s)	Name	Description	State/Range
31 - 25	Reserved	-	0
24	peri_int	This bit is used to acknowledge a peripheral interrupt. This must be done to enable new interrupts from a peripheral. To avoid false interrupts, it should also be done before peri_int is enabled for the first time. This bit takes action on write, and its state is not saved.	0 = nop 1 = ack
23 - 21	Reserved	-	0
20	oe	This bit enables or disables the output for the data buffer when Parallel Port p0 is in manual mode.	0 = disable 1 = enable
19	seli	This bit controls the $\overline{\text{p0selectin}}$ signal when Parallel Port p0 is in manual mode. $\overline{\text{p0selectin}}$ is asserted when the seli bit is set to active. (note 1)	0 = inactive 1 = active

18	autofd	This bit controls the p0autofd signal when Parallel Port p0 is in Manual, Centronics (Compatibility), or IBM fastbyte modes. p0autofd is asserted when the autofd bit is set to active. (note 2)	0 = inactive 1 = active
17	strb	This bit controls the p0strobe signal when Parallel Port p0 is in manual mode. p0strobe is asserted when the strb bit is set to active. (note 3)	0 = inactive 1 = active
16	init	This bit controls the p0init signal when Parallel Port p0 is in manual, Centronics (Compatibility), IBM fastbyte, nibble or byte modes. p0init is asserted when the init bit is set to active. (note 4)	0 = inactive 1 = active
15 - 9	Reserved	-	0
8	ecp_cmd	If Parallel Port p0 is in ECP forward mode, this bit indicates whether the transmitted byte contains data or a command. The bit is automatically set low when a command byte has been transmitted.	0 = data 1 = command
7 - 0	data	Output data to p0d7 - p0d0 in non DMA operation.	

Note 1: The polarity of the p0selectin signal depends upon inversion control bit control bit 24 (iseli) in register R_PAR0_CONFIG. When iseli is set to noninv (0), the value active will set p0selectin to 0.

Note 2: The polarity of the p0autofd signal depends upon inversion control bit 23 (iautofd) in register R_PAR0_CONFIG. When iautofd is set to noninv (0), the value active will set p0autofd to 0.

Note 3: The polarity of the p0strobe signal depends upon inversion control bit 22 (istrb) in register R_PAR0_CONFIG. When istrb is set to noninv (0), the value active will set p0strobe to 0.

Note 4: The polarity of the p0init signal depends upon inversion control bit 21 (iinit) in register R_PAR0_CONFIG. When iinit is set to noninv (0), the value active will set p0init to 0.

18.10.2 R_PAR0_CTRL

Parallel Port p0 Control Register, General Characteristics

ID of register	R_PAR0_CTRL	**Size**	8 bits
Offset	0x42	**Read/Write**	Write only
Register address	0xB0000042	**Initial value**	Unknown

Bit Assignments of R_PAR0_CTRL

Bit(s)	Name	Description	State/Range
7 - 5	Reserved	-	0
4 - 0	ctrl	This field is the same as bits 20 to 16 in register R_PAR0_CTRL_DATA.	0 - 31

18.10.3 R_PAR0_STATUS_DATA

Parallel Port p0 Status and Data Register, General Characteristics

ID of register	R_PAR0_STATUS_DATA	**Size**	32 bits
Offset	0x40	**Read/Write**	Read only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_PAR0_STATUS_DATA

Bit(s)	Name	Description	State/Range
31 - 29	mode	The value of this field indicates the current data transfer mode status of Parallel Port p0. These values are valid only if ext_mode in register "R_PAR0_CONFIG" is set. (note 1)	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off 0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3
28	perr	This bit indicates the status of external peripheral error signal p0perror at Parallel Port p0. (note 2)	0 = inactive 1 = active
27	ack	This bit indicates the status of external acknowledge signal $\overline{\text{p0ack}}$ at Parallel Port p0. (note 2)	0 = active 1 = inactive
26	busy	This bit indicates the status of external signal p0busy at Parallel Port p0. (note 2)	0 = inactive 1 = active
25	fault	This bit indicates the status of external signal $\overline{\text{p0fault}}$ at Parallel Port p0. (note 2)	0 = active 1 = inactive
24	sel	This bit indicates the status of external signal p0select at Parallel Port p0. (note 2)	0 = inactive 1 = active
23	ext_mode	This bit indicates whether EPP extended mode is selected at Parallel Port p0. (note 3)	0 = disable 1 = enable
22	ecp_16	This bit indicates whether ECP 16-bit mode is selected at Parallel Port p0.	0 = inactive 1 = active
21 - 18	Reserved	-	0
17	tr_rdy	This bit indicates whether a peripheral is ready for a transmission from Parallel Port p0. It is set ready (1) when a new byte can be written to register "R_PAR0_CTRL_DATA".	0 = busy 1 = ready
16	dav	This bit indicates whether new data has been received in register "R_PAR0_STATUS_DATA" in non DMA mode. In DMA mode, set this bit to data (1) whenever data is stored in the DMA FIFO. In both DMA and non DMA modes this bit is cleared when R_PAR0_STATUS_DATA is read.	0 = nodata 1 = data
15 - 9	Reserved	-	0

Bit Assignments of R_PAR0_STATUS_DATA (continued)

8	ecp_cmd	If Parallel Port p0 is in ECP reverse mode, bit ecp_cmd indicates whether the received byte contains data or a command.	0 = data 1 = command
7 - 0	data	Contains the latest byte received on the data lines at Parallel Port p0.	

Note 1: This field can be used to detect mode change when force in register R_PAR0_CONFIG is set to off, see the mode field in R_PAR0_CONFIG.

Note 2: The state of the p0ack, p0busy, p0fault, p0perror and p0select signals depend respectively upon the states of iack, ibusy, ifault, iperr and isel in register R_PAR0_CONFIG.The table below gives the relationship between pin values, inversion control bits, and status bits:

p0ack p0busy p0perror p0fault p0select	iack ibusy ifault iperr isel	ack busy fault perr sel
0	0	0
1	0	1
0	1	1
1	1	0

Note 3: See 18.10.6, note 2.

18.10.4 R_PAR0_STATUS

Parallel Port p0 Status Register, General Characteristics

ID of register	R_PAR0_STATUS	**Size**	16 bits (Word)
Offset	0x42	**Read/Write**	Read only
Register address	0xB0000042	**Initial value**	Unknown

Bit Assignments of R_PAR0_STATUS

Bit(s)	Name	Description	State/Range
15 - 13	mode	The value of this field indicates the current data transfer mode of Parallel Port p0. These values are valid only if ext_mode in register "R_PAR0_CONFIG" is set. (note 1)	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off 0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3
12	perr	This bit indicates the status of external peripheral error signal p0perror at Parallel Port p0. (note 2)	0 = inactive 1 = active
11	ack	This bit indicates the status of external acknowledge signal $\overline{\text{p0ack}}$ at Parallel Port p0. (note 3)	0 = active 1 = inactive
10	busy	This bit indicates the status of external signal p0busy at Parallel Port p0. (note 3)	0 = inactive 1 = active
9	fault	This bit indicates the status of external signal $\overline{\text{p0fault}}$ at Parallel Port p0. (note 3)	0 = active 1 = inactive
8	sel	This bit indicates the status of external signal p0select at Parallel Port p0. (note 3)	0 = inactive 1 = active
7	ext_mode	This bit indicates whether the EPP extended mode is selected at Parallel Port p0. (note 4)	0 = disable 1 = enable
6	ecp_16	This bit indicates whether the ECP 16-bit mode is selected at Parallel Port p0.	0 = inactive 1 = active
5 - 2	Reserved	-	0
1	tr_rdy	This bit indicates whether a peripheral is ready for a transmission from Parallel Port p0. It is set ready (1) when a new byte can be written to register "R_PAR0_CTRL_DATA".	0 = busy 1 = ready
0	dav	This bit indicates whether new data has been received in register "R_PAR0_STATUS_DATA" in non DMA mode. In DMA mode, set this bit to data (1) whenever data is stored in the DMA FIFO. In both DMA and non DMA modes this bit is cleared when R_PAR0_STATUS_DATA is read.	0 = nodata 1 = data

Note 1: This field can be used to detect mode change when force in register R_PAR0_CONFIG is set to off, see the mode field in R_PAR0_CONFIG.

Note 2: The state of the p0ack, p0busy, p0fault, p0perror and p0select signals depend respectively upon the states of iack, ibusy, ifault, iperr and isel in register R_PAR0_CONFIG.The table below gives the relationship between pin values, inversion control bits, and status bits:

p0ack p0busy p0perror p0fault p0select	iack ibusy ifault iperr isel	ack busy fault perr sel
0	0	0
1	0	1
0	1	1
1	1	0

Note 3: See 18.10.6, note 2.

Note 4: This 16-bit register is part of the 32-bit register “R_PAR0_STATUS_DATA”.

18.10.5 R_PAR_ECP16_DATA

Parallel Port ECP Wide Data Register, General Characteristics

ID of register	R_PAR_ECP16_DATA	**Size**	16 bits
Offset	0x40	**Read/Write**	Read/Write
Address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_PAR_ECP16_DATA

Bit(s)	Name	Description	State/Range
15 - 0	data	In ECP wide mode this register contains the last data word sent or received.	

18.10.6 R_PAR0_CONFIG

Parallel Port p0 Configuration Register, General Characteristics

ID of register	R_PAR0_CONFIG	**Size**	32 bits
Offset	0x44	**Read/Write**	Write only
Address	0xB0000044	**Initial value**	0

Bit Assignments of R_PAR0_CONFIG

Bit(s)	Name	Description	State/Range
31 - 26	Reserved	-	0
25	ioe	This bit inverts the data output enable signal. It is set if the output enable signal on the (optional) external driver is active low.	0 = noninv 1 = inv
24	iseli	This bit inverts the $\overline{\text{p0selectin}}$ signal. In manual mode, if seli in R_PAR0_CTRL_DATA is set to active (1), the $\overline{\text{p0selectin}}$ pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
23	iautofd	This bit inverts the $\overline{\text{p0autofd}}$ signal. In manual mode, if autofd in R_PAR0_CTRL_DATA is set to active (1), the $\overline{\text{p0autofd}}$ pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
22	istrb	This bit inverts the $\overline{\text{p0strobe}}$ signal. In manual mode, if strb in R_PAR0_CTRL_DATA is set to active (1), the $\overline{\text{p0strobe}}$ pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
21	iinit	This bit inverts the $\overline{\text{p0init}}$ signal. In manual mode, if init in R_PAR0_CTRL_DATA is set to active (1), the $\overline{\text{p0init}}$ pin will be set to one if this bit is set to inv (1).	0 = non-invert 1 = invert
20	iperr	This bit inverts the p0perror signal. If this bit is set to inv (1) and the p0perror signal is set to one, perr in R_PAR0_STATUS is set to inactive (0).	0 = noninv 1 = inv
19	iack	This bit inverts the $\overline{\text{p0ack}}$ signal. If this bit is set to inv (1) and the $\overline{\text{p0ack}}$ signal is set to one, ack in R_PAR0_STATUS is set to active (0).	0 = noninv 1 = inv
18	ibusy	This bit inverts the p0busy signal. If this bit is set to inv (1) and the p0busy signal is set to one, busy in R_PAR0_STATUS is set to inactive (0).	0 = noninv 1 = inv
17	ifault	This bit inverts the $\overline{\text{p0fault}}$ signal. If this bit is set to inv (1) and the $\overline{\text{p0fault}}$ signal is set to one, fault in R_PAR0_STATUS is set to active (0).	0 = noninv 1 = inv
16	isel	This bit inverts the p0select signal. If this bit is set to inv (1) and the p0select signal is set to one, sel in R_PAR0_STATUS is set to inactive (0).	0 = noninv 1 = inv
15 - 12	Reserved	-	0
11	ext_mode	This bit enables the EPP read and write modes in parallel Port p0. (note 2)	0 = disable 1 = enable
10	wide	This bit enables the ECP 16-bit mode in Parallel Port p0. (note 3)	0 = disable 1 = enable
9	dma	This bit enables the DMA operation for Parallel Port p0.	0 = disable 1 = enable

Bit Assignments of R_PAR0_CONFIG (continued)

Bit(s)	Name	Description	State/Range
8	rle_in	When Parallel Port p0 is in ECP mode, this bit enables the expansion of RLE-compressed incoming data.	0 = disable 1 = enable
7	rle_out	When Parallel Port p0 is in ECP mode, this bit enables the RLE compression of outgoing data.	0 = disable 1 = enable
6	enable	This bit enables/resets Parallel Port p0.	0 = reset 1 = on
5	force	When on, this bit forces an immediate mode change in Parallel Port p0. When off, the mode changes at the next handshake. To detect a mode change, register R_PAR0_STATUS must be read.	0 = off 1 = on
4	ign_ack	When Parallel Port p0 is in (compatibility) mode and this bit is set to ignore (1), the p0ack signal is ignored. (note 4)	0 = wait 1 = ignore
3	oe_ack epp_addr_data	This bit controls whether data output is disabled after the hold time set in R_PAR0_DELAY has expired, or if data output is enabled until an ack is received. When Parallel Port p0 is in Centronics (Compatibility) mode, the bit determines whether the port waits, or does not wait, for acknowledgement from the peripheral. (note 5) When Parallel Port p0 is in EPP mode, the bit indicates whether the byte present at Parallel Port p0 contains data or an address.	 0 = dont_wait 1 = wait_oe 0 = epp_data 1 = epp_addr
2 - 0	mode	The value of this field selects the current data transfer mode of Parallel Port p0. The OFF mode can be used to initiate an IEEE 1284 termination phase. In OFF mode the output signals of Parallel Port p0 are set as follows: p0selectin = active (0) (IEEE 1284 not active) p0autofd = inactive (1) p0strobe = inactive (1) p0init = inactive (1) (note 1) To operate in an EPP mode, ext_mode in this register must be set to enable. (note 2)	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off 0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3

Note 1: The values, in parentheses, for the physical output signals above are affected by the value of the inversion bits in R_PAR0_CTRL_DATA.

Note 2: The EPP mode is selected by the ext_mode field and bits 2 to 0 of the mode field in register R_PAR0_CONFIG. The ext_mode field is the mode on/off switch and bits 2 to 0 of the mode field choose the read or write mode. This is shown in the truth table below.

ext_mode	mode			EPP Mode
	bit 2	bit 1	bit 0	
0	-	-	-	EPP mode off
1	0	0	0	EPP read mode
1	1	0	1	EPP write mode 1
1	1	1	0	EPP write mode 2
1	1	1	1	EPP write mode 3

Note 3: To enable the ECP wide mode in Parallel Port p0, par_w and par0 in general configuration register R_GEN_CONFIG must both be set to select.

Note 4: To obtain the expected result when setting the ign_ack bit to ignore, oe_ack must be set to dont_wait.

Note 5: To obtain the expected result when setting the oe_ack bit to wait_oe, ign_ack must be set to wait.

18.10.7 R_PAR0_DELAY

Parallel Port p0 Delay Register, General Characteristics

ID of register	R_PAR0_DELAY	**Size**	32 bits (Dword)
Offset	0x48	**Read/Write**	Write only
Register address	0xB0000048	**Initial value**	Unknown

Bit Assignments of R_PAR0_DELAY

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23 - 21	fine_hold	The value of this field applies fine adjustment to the hold time (T_{hold}) of Parallel Port p0.	0 -7
20 - 16	hold	The value of this field determines the minimum hold time (T_{hold}) when Parallel Port p0 is in Centronics (Compatibility) mode. It is the period from the active to inactive transition of the $\overline{\text{p0strobe}}$ signal until p0d7 to p0d0 are no longer valid. If the oe_ack bit in R_PAR0_CONFIG is used, the data is valid until ack arrives. (note 1)	0 - 31 (20 ns to 5.0 µs)
15 - 13	fine_strb	The value of this field applies fine adjustment to the strobe time (T_{strb}) of Parallel Port p0. (note 2)	0 - 7
12 - 8	strobe	The value of this field determines the strobe time (T_{strb}) when Parallel Port p0 is in Centronics (Compatibility) mode. It is the period in which the $\overline{\text{p0strobe}}$ signal is active. (note 2)	0 - 31 (20 ns to 5.0 µs)
7 - 5	fine_setup	The value of this field applies fine adjustment to the setup time (T_{ub}) of Parallel Port p0. (note 3)	0 - 7
4 - 0	setup	The value of this field determines the setup time (T_{su}) when Parallel Port p0 is in IBM fastbyte, Centronics (Compatibility) and ECP forward modes. It is the period from the start of data output to the inactive to active transition of the $\overline{\text{p0strobe}}$ signal. (note 3)	0 - 31 (10 ns to 5.0 µs)

Note 1: The formula for T_{hold} is (hold x 160 + fine_hold x 20 + 20) ns.

Note 2: The formula for T_{srtb} is (strobe x 160 + fine_strb x 20 + 20) ns.

Note 3: The formula for T_{su} is (setup x 160 + fine_setup x 20 + 10) ns. For example if setup is binary 00010 (0x2) and fine_setup is binary 001 (0x1), then T_{su} will be (2 x 160) + (1 x 20) + 10 = 350 ns.

18.10.8 R_PAR1_CTRL_DATA

Parallel Port p1 Control and Data Register, General Characteristics

ID of register	R_PAR1_CTRL_DATA	Size	32 bits (Dword)
Offset	0x50	Read/Write	Write only
Register address	0xB0000050	Initial value	Unknown

Bit Assignments of R_PAR1_CTRL_DATA

Bit(s)	Name	Description	State/Range
31 - 25	Reserved	-	0
24	peri_int	This bit is used to acknowledge a peripheral interrupt. This must be done to enable new interrupts from a peripheral. To avoid false interrupts, it should also be done before peri_int is enabled for the first time. This bit takes action on write, and its state is not saved.	0 = nop 1 = ack
23 - 21	Reserved	-	0
20	oe	This bit enables or disables the output for the data buffer when Parallel Port p1 is in manual mode.	0 = disable 1 = enable
19	seli	This bit controls the $\overline{\text{p1selectin}}$ signal when Parallel Port p1 is in manual mode. $\overline{\text{p1selectin}}$ is asserted when the seli bit is set (active). (note 1)	0 = inactive 1 = active
18	autofd	This bit controls the $\overline{\text{p1autofd}}$ signal when Parallel Port p1 is in manual, Centronics (Compatibility), or IBM fastbyte modes. $\overline{\text{p1autofd}}$ is asserted when the autofd bit is set (active). (note 2)	0 = inactive 1 = active
17	strb	This bit controls the $\overline{\text{p1strobe}}$ signal when Parallel Port p1 is in manual mode. $\overline{\text{p1strobe}}$ is asserted when the strb bit is set (active). (note 3)	0 = inactive 1 = active
16	init	This bit controls the $\overline{\text{p1init}}$ signal when Parallel Port p1 is in manual, Centronics (Compatibility), IBM fastbyte, nibble or byte modes. $\overline{\text{p1init}}$ is asserted when the init bit is set (active). (note 4)	0 = inactive 1 = active
15 - 9	Reserved	-	0
8	ecp_cmd	If Parallel Port p1 is in ECP forward mode, this bit indicates whether the transmitted byte contains data or a command.	0 = data 1 = command
7 - 0	data	Data byte to p1d7 - p1d0 in non DMA operation.	

Note 1: The polarity of the $\overline{\text{p1selectin}}$ signal depends upon inversion control bit control bit 24 (iseli) in register R_PAR1_CONFIG. When iseli is set to noninv (0), the value active will set $\overline{\text{p1selectin}}$ to 0.

Note 2: The polarity of the $\overline{\text{p1autofd}}$ signal depends upon inversion control bit 23 (iautofd) in register R_PAR1_CONFIG. When iautofd is set to noninv (0), the value active will set $\overline{\text{p1autofd}}$ to 0.

Note 3: The polarity of the $\overline{\text{p1strobe}}$ signal depends upon inversion control bit 22 (istrb) in register R_PAR1_CONFIG. When istrb is set to noninv (0), the value active will set $\overline{\text{p1strobe}}$ to 0.

Note 4: The polarity of the $\overline{\text{p1init}}$ signal depends upon inversion control bit 21 (iinit) in register

R_PAR1_CONFIG. When iinit is set to noninv (0), the value active will set $\overline{\text{p1init}}$ to 0.

18.10.9 R_PAR1_CTRL

Parallel Port p1 Control Register, General Characteristics

ID of register	R_PAR1_CTRL	**Size**	8 bits
Offset	0x52	**Read/Write**	Write only
Register address	0xB0000052	**Initial value**	Unknown

Bit Assignments of R_PAR1_CTRL

Bit(s)	Name	Description	State/Range
7 - 5	Reserved	-	0
4 - 0	ctrl	This field is the same as bits 20 to 16 in register “R_PAR1_CTRL_DATA”.	0 - 31

18.10.10 R_PAR1_STATUS_DATA

Parallel Port p1 Status and Data Register, General Characteristics

ID of register	R_PAR1_STATUS_DATA	**Size**	32 bits
Offset	0x50	**Read/Write**	Read only
Register address	0xB0000050	**Initial value**	Unknown

Bit Assignments of R_PAR1_STATUS_DATA

Bit(s)	Name	Description	State/Range
31 - 29	mode	The value of this field indicates the current data transfer mode status of Parallel Port p1.	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off
		These values are valid only if ext_mode in register “R_PAR1_CONFIG” is set. (note 1)	0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3
28	perr	This bit indicates the status of external peripheral error signal p1perror at Parallel Port p1. (note 2)	0 = inactive 1 = active
27	ack	This bit indicates the status of external acknowledge signal $\overline{\text{p1ack}}$ at Parallel Port p1. (note 2)	0 = active 1 = inactive
26	busy	This bit indicates the status of external signal p1busy at Parallel Port p1. (note 2)	0 = inactive 1 = active
25	fault	This bit indicates the status of external signal $\overline{\text{p1fault}}$ at Parallel Port p1. (note 2)	0 = active 1 = inactive
24	sel	This bit indicates the status of external signal p1select at Parallel Port p1. (note 2)	0 = inactive 1 = active
23	ext_mode	This bit indicates whether EPP extended mode is selected at Parallel Port p1. (note 3)	0 = disable 1 = enable
22	ecp_16	This bit indicates whether ECP 16-bit mode is selected at Parallel Port p1.	0 = inactive 1 = active
21 - 18	Reserved	-	0
17	tr_rdy	This bit indicates whether a peripheral is ready for a transmission from Parallel Port p0. It is set ready (1) when a new byte can be written to register “R_PAR1_CTRL_DATA”.	0 = busy 1 = ready
16	dav	This bit indicates whether new data has been received in register “R_PAR1_STATUS_DATA” in non DMA mode. In DMA mode, set this bit to data (1) whenever data is stored in the DMA FIFO. In both DMA and non DMA modes this bit is cleared when “R_PAR1_STATUS_DATA” is read.	0 = nodata 1 = data
15 - 9	Reserved	-	0

Bit Assignments of R_PAR1_STATUS_DATA (continued)

8	ecp_cmd	If Parallel Port p1 is in ECP reverse mode, bit ecp_cmd indicates whether the received byte contains data or a command.	0 = data 1 = command
7 - 0	data	Contains the latest byte received on data lines d1d7:d1d0 at Parallel Port p1.	

Note 1: This field can be used to detect mode change when force in register R_PAR1_CONFIG is set to off, see the mode field in R_PAR1_CONFIG.

Note 2: The state of the p1ack, p1busy, p1perror, p1fault and p1select signals depend respectively upon the states of iack, ibusy, ifault, iperr and isel in register R_PAR1_CONFIG.The table below gives the relationship between pin values, inversion control bits, and status bits:

p1ack p1busy p1perror p1fault p1select	iack ibusy ifault iperr isel	ack busy fault perr sel
0	0	0
1	0	1
0	1	1
1	1	0

Note 3: See 18.10.12, note 2.

18.10.11 R_PAR1_STATUS

Parallel Port p1 Status Register, General Characteristics

ID of register	R_PAR1_STATUS	**Size**	16 bits
Offset	0x52	**Read/Write**	Read only
Register address	0xB0000052	**Initial value**	Unknown

Bit Assignments of R_PAR1_STATUS

Bit(s)	Name	Description	State/Range
15 - 13	mode	The value of this field indicates the current data transfer mode of Parallel Port p1. These values are valid only if bit 11 in register “R_PAR1_CONFIG” is set. (note 1)	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off 0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3
12	perr	This bit indicates the status of external peripheral error signal p1perror at Parallel Port p1. (note 2)	0 = inactive 1 = active
11	ack	This bit indicates the status of external acknowledge signal $\overline{\text{p1ack}}$ at Parallel Port p1. (note 3)	0 = active 1 = inactive
10	busy	This bit indicates the status of external signal p1busy at Parallel Port p1. (note 3)	0 = inactive 1 = active
9	fault	This bit indicates the status of external signal $\overline{\text{p1fault}}$ at Parallel Port p1. (note 3)	0 = active 1 = inactive
8	sel	This bit indicates the status of external signal p1select at Parallel Port p1. (note 3)	0 = inactive 1 = active
7	ext_mode	This bit indicates whether EPP extended mode is selected at Parallel Port p1. (note 4)	0 = disable 1 = enable
6 - 2	Reserved	-	0
1	tr_rdy	This bit indicates whether a peripheral is ready for a transmission from Parallel Port p0. It is set ready (1) when a new byte can be written to register “R_PAR1_CTRL_DATA”.	0 = busy 1 = ready
0	dav	This bit indicates whether new data has been received in register “R_PAR1_STATUS_DATA” in non DMA mode. In DMA mode, set this bit to data (1) whenever data is stored in the DMA FIFO. In both DMA and non DMA modes this bit is cleared when “R_PAR1_STATUS_DATA” is read.	0 = nodata 1 = data

Note 1: This field can be used to detect mode change when force in register R_PAR1_CONFIG is set to off, see the mode field in R_PAR1_CONFIG.

Note 2: The state of the $\overline{\text{p1ack}}$, p1busy, p1perror, $\overline{\text{p1fault}}$ and p1select signals depend respectively upon the states of iack, ibusy, ifault, iperr and isel in register R_PAR1_CONFIG.The table below gives the relationship between pin values, inversion control bits, and status bits:

$\overline{\text{p1ack}}$ p1busy p1perror $\overline{\text{p1fault}}$ p1select	iack ibusy ifault iperr isel	ack busy fault perr sel
0	0	0
1	0	1
0	1	1
1	1	0

Note 3: See 18.10.12, note 2.

Note 4: This 16-bit register is part of the 32-bit register “R_PAR1_STATUS_DATA”.

18.10.12 R_PAR1_CONFIG

Parallel Port p1 Configuration Register, General Characteristics

ID of register	R_PAR1_CONFIG	**Size**	32 bits
Offset	0x54	**Read/Write**	Write only
Address	0xB0000054	**Initial value**	0

Bit Assignments of R_PAR1_CONFIG

Bit(s)	Name	Description	State/Range
31 - 26	Reserved	-	0
25	ioe	This bit inverts the data output enable signal. It is set if the output enable signal on the (optional) external driver is active low.	0 = noninv 1 = inv
24	iseli	This bit inverts the p1selectin signal. In manual mode, if seli in R_PAR1_CTRL_DATA is set to active (1), the p1selectin pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
23	iautofd	This bit inverts the p1autofd signal. In manual mode, if autofd in R_PAR1_CTRL_DATA is set to active (1), the p1autofd pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
22	istrb	This bit inverts the p1strobe signal. In manual mode, if strb in R_PAR1_CTRL_DATA is set to active (1), the p1strobe pin will be set to one if this bit is set to inv (1).	0 = noninv 1 = inv
21	iinit	This bit inverts the p1init signal. In manual mode, if init in R_PAR1_CTRL_DATA is set to active (1), the p1init pin will be set to one if this bit is set to inv (1).	0 = non-invert 1 = invert
20	iperr	This bit inverts the p1perror signal. If this bit is set to inv (1) and the p1perror signal is set to one, perr in R_PAR1_STATUS is set to inactive (0).	0 = noninv 1 = inv
19	iack	This bit inverts the p1ack signal. If this bit is set to inv (1) and the p1ack signal is set to one, ack in R_PAR1_STATUS is set to active (0).	0 = noninv 1 = inv
18	ibusy	This bit inverts the p1busy signal. If this bit is set to inv (1) and the p1busy signal is set to one, busy in R_PAR1_STATUS is set to inactive (0).	0 = noninv 1 = inv
17	ifault	This bit inverts the p1fault signal. Iff this bit is set to inv (1) and the p1fault signal is set to one, fault in R_PAR1_STATUS is set to active (0).	0 = noninv 1 = inv
16	isel	This bit inverts the p1select signal. If this bit is set to inv (1) and the p1select signal is set to one, sel in R_PAR1_STATUS is set to inactive (0).	0 = noninv 1 = inv
15 - 12	Reserved	-	0
11	ext_mode	This bit enables the EPP read and write modes in parallel Port p1. (note 2)	0 = disable 1 = enable
10	Reserved	-	0
9	dma	This bit enables the DMA operation for Parallel Port p1.	0 = disable 1 = enable

Bit Assignments of R_PAR1_CONFIG (continued)

Bit(s)	Name	Description	State/Range
8	rle_in	When Parallel Port p1 is in ECP mode, this bit enables the expansion of RLE-compressed incoming data.	0 = disable 1 = enable
7	rle_out	When Parallel Port p1 is in ECP mode, this bit enables the RLE compression of outgoing data.	0 = disable 1 = enable
6	enable	This bit enables/resets Parallel Port p1.	0 = reset 1 = on
5	force	When on (1), this bit forces an immediate mode change in Parallel Port p1. When off (0), the mode changes at the next handshake. To detect a mode change, register R_PAR1_STATUS must be read.	0 = off 1 = on
4	ign_ack	When Parallel Port p1 is in Centronics (Compatibility) mode and this bit is set to ignore (1), the p1ack signal is ignored. (note 3)	0 = wait 1 = ignore
3	oe_ack epp_addr_data	This bit controls whether data output is disabled after the hold time set in R_PAR1_DELAY has expired, or if data output is enabled until an ack is received. When Parallel Port p1 is in Centronics (Compatibility) mode, this bit determines whether the port waits, or does not wait, for acknowledgement from the peripheral. (note 4) When Parallel Port p1 is in EPP mode, the bit indicates whether the byte contains data or an address.	 0 = dont_wait 1 = wait_oe 0 = epp_data 1 = epp_addr
2 - 0	mode	The value of this field selects the current data transfer mode of Parallel Port p1. The OFF mode can be used to initiate an IEEE 1284 termination phase. In OFF mode the output signals of Parallel Port p1 are set as follows: p1selectin = active (0) (IEEE 1284 not active) p1autofd = inactive (1) p1strobe = inactive (1) p1init = inactive (1) (note 1) To operate in an EPP mode, ext_mode in this register must be set to enable. (note 2)	0 = manual 1 = centronics 2 = fastbyte 3 = nibble 4 = byte 5 = ecp_fwd 6 = ecp_rev 7 = off 0 = epp_rd 5 = epp_wr1 6 = epp_wr2 7 = epp_wr3

Note 1: The values, in parentheses, for the physical output signals above are affected by the value of the inversion bits in R_PAR1_CTRL_DATA.

Note 2: The EPP mode is selected by the ext_mode field and bits 2 to 0 of the mode field in register R_PAR1_CONFIG. The ext_mode field is the mode on/off switch and bits 2 to 0 of the mode field choose the read or write mode. This is shown in the truth table below.

ext_mode	mode			EPP Mode
	bit 2	bit 1	bit 0	
0	-	-	-	EPP mode off
1	0	0	0	EPP read mode
1	1	0	1	EPP write mode 1
1	1	1	0	EPP write mode 2
1	1	1	1	EPP write mode 3

Note 3: To obtain the expected result when setting the ign_ack bit to ignore, oe_ack must be set to dont_wait.

Note 4: To obtain the expected result when setting the oe_ack bit to wait_oe, ign_ack must be set to wait.

18.10.13 R_PAR1_DELAY

Parallel Port p1 Delay Register, General Characteristics

ID of register	R_PAR1_DELAY	**Size**	32 bits
Offset	0x58	**Read/Write**	Write only
Register address	0xB0000058	**Initial value**	Unknown

Bit Assignments of R_PAR1_DELAY

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23 - 21	fine_hold	The value of this field applies fine adjustment to the hold time (T_{hold}) of Parallel Port p1. (note 1)	0 -7
20 - 16	hold	The value of this field determines the minimum hold time (T_{hold}) when Parallel Port p0 is in Centronics (Compatibility) mode. It is the period from the active to inactive transition of the p1strobe signal until p1d7 to p1d0 are no longer valid. If the oe_ack bit in R_PAR1_CONFIG is used, the data is valid until ack arrives. (note 1)	0 - 31 (20 ns to 5.1 µs)
15 - 13	fine_strb	The value of these bits applies fine adjustment to the strobe time (T_{strb}) of Parallel Port p1. (note 2)	0 - 7
12 - 8	strobe	The value of these bits determines the strobe time (T_{strb}) when Parallel Port p1 is in Centronics (Compatibility) mode. It is the period in which the p1strobe signal is active. (note 2)	0 - 31 (20 ns to 5.1 µs)
7 - 5	fine_setup	The value of these bits applies fine adjustment to the setup time (T_{ub}) of Parallel Port p1. (note 3)	0 - 7
4 - 0	setup	The value of these bits determines the setup time (T_{su}) when Parallel Port p1 is in IBM Fastbyte, Centronics (Compatibility) and ECP forward modes. It is the period from the start of data output to the inactive to active transition of the p1strobe signal. (note 3)	0 - 31 (10 ns to 5.0 µs)

Note 1: The formula for T_{hold} is (hold x 160 + fine_hold x 20 + 20) ns.

Note 2: The formula for T_{srtb} is (strobe x 160 + fine_strb x 20 + 20) ns.

Note 3: The formula for T_{su} is (setup x 160 + fine_setup x 20 + 10) ns. For example if setup is binary 00010 (0x2) and fine_setup is binary 001 (0x1), then T_{su} will be (2 x 160) + (1 x 20) + 10 = 350 ns.

18.11 ATA Interface Registers

18.11.1 R_ATA_CTRL_DATA

ATA Control and Data Register, General Characteristics

ID of register	R_ATA_CTRL_DATA	**Size**	32 bits
Offset	0x40	**Read/Write**	Write only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_ATA_CTRL_DATA

Bit(s)	Name	Description	State/Range
31 - 30	sel	This field selects ATA bus 0 - 3.	0 - 3
29	cs1	This bit activates chip select 1 on the ATA bus. cs1 and cs0 should never be set to the same value, either to inactive (0) or active (1), at the same time.	0 = inactive 1 = active
28	cs0	This bit activates chip select 0 on the ATA bus. cs0 and cs1 should never be set to the same value, either to inactive (0) or active (1), at the same time.	0 = inactive 1 = active
27 - 25	addr	This field gives the ATA bus device address.	0 - 7
24	rw	This bit selects read or write.	0 = write 1 = read
23	src_dst	If this bit is set to dma (1), the source (write) or the destination (read) for data is the internal DMA. Otherwise the data field in this register is the source and the data field in “R_ATA_STATUS_DATA” is the destination.	0 = register 1 = dma
22	handsh	This bit selects PIO or DMA handshaking on the ATA bus.DMA handshaking can only be used for data transfers with ATA DMA commands.	0 = pio 1 = dma
21	multi	If this bit is set to on (1), transfers, each consisting of one PIO read and write or one DMA burst, will continue whenever there are data to be read or written. If the bit is not set (off), this register has to be written again to continue. Bit 18 (busy) in “R_ATA_STATUS_DATA” will be set to yes until “R_ATA_TRANSFER_CNT” reaches zero.	0 = off 1 = on
20	dma_size	This bit selects the size of the DMA transfers, 16 bits (word) or 8 bits (byte). Byte transfers are only needed for devices that use 8 bit transfers to/from the data register and devices that do not support DMA handshaking.	0 = word 1 = byte
19 - 16	Reserved	-	0
15 - 0	data	The data in this field is written to the ATA unit if src_dst is set to register and rw is set to write.	

Note: This register is used to read and write data to and from ATA units. When writing, data is supplied by the data field in this register or by the internal DMA. When reading, data ends up in the data field in R_ATA_STATUS_DATA register or in the internal DMA. Writing to the register starts a transfer. Before a new read or write is written to this register, busy in R_ATA_STATUS_DATA should be read as cleared no (0). If not, the current transfer will be interrupted.

18.11.2 R_ATA_STATUS_DATA

ATA Status and Data Register, General Characteristics

ID of register	R_ATA_STATUS_DATA	**Size**	32 bits
Offset	0x40	**Read/Write**	Read only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_ATA_STATUS_DATA

Bit(s)	Name	Description	State/Range
31 - 19	Reserved	-	0
18	busy	This bit indicates if the ATA interface is busy or not. If it is set to yes (1), when writing a command to “R_ATA_CTRL_DATA”, the command in progress will be interrupted. When the multi bit in “R_ATA_CTRL_DATA” is set to on (1), this bit will remain set yes (1) until “R_ATA_TRANSFER_CNT” reaches zero.	0 = no 1 = yes
17	tr_rdy	This bit indicates if the transmitter is ready. When it is set (ready), it is possible to write new data to the data field in “R_ATA_CTRL_DATA”. The bit is not valid when using the internal DMA.	0 = busy 1 = ready
16	dav	If this bit is set (data), there is new data available in the data field of this register. The bit is cleared when the register is read. It is not valid when using the internal DMA.	0 = nodata 1 = data
15 - 0	data	This field contains data read from the ATA unit if both src_dst is set to register and rw is set to read, in “R_ATA_CTRL_DATA”.	

18.11.3 R_ATA_CONFIG

ATA Configuration Register, General Characteristics

ID of register	R_ATA_CONFIG	**Size**	32 bits
Offset	0x44	**Read/Write**	Write only
Register address	0xB0000044	**Initial value**	Unknown

Bit Assignments of R_ATA_CONFIG

Bit(s)	Name	Description	State/Range
31 - 26	Reserved	-	0
25	enable	This bit enables the ATA controller.	0 = off 1 = on
24 - 20	dma_strobe	This field determines the strobe time for DMA handshaking.	0 - 31
19 - 15	dma_hold	This field determines the hold time for DMA handshaking.	0 - 31
14 - 10	pio_setup	This field determines the setup time for PIO read and writes.	0 - 31
9 - 5	pio_strobe	This field determines the strobe time for PIO read and writes.	0 - 31
4 - 0	pio_hold	This field determines the hold time for PIO read and writes.	0 - 31

Note: Times are calculated as (r + 1)*20ns where r is the register value for a specific time.

18.11.4 R_ATA_TRANSFER_CNT

ATA Transfer Count Register, General Characteristics

ID of register	R_ATA_TRANSFER_CNT	**Size**	32 bits
Offset	0x48	**Read/Write**	Read/Write
Register address	0xB0000048	**Initial value**	Unknown

Bit Assignments of R_ATA_TRANSFER_CNT

Bit(s)	Name	Description	State/Range
31 - 17	Reserved	-	0
16 - 0	count	This counter is decremented each time a 16 bit word (or a byte if dma_size in “R_ATA_CTRL_DATA” is set to byte) is transferred to or from an ATA device. It is primarily used when transferring data using the internal DMA. When the counter reaches zero, DMA transfers are stopped and an EOP is generated. When multi in “R_ATA_CTRL_DATA” is set to on, busy in “R_ATA_STATUS_DATA” will remain set to yes until this counter reaches zero.	0 - 131071

18.12 SCSI Registers

18.12.1 R_SCSI0_CTRL

SCSI-8 p0 and SCSI-W Control Register, General Characteristics

ID of register	R_SCSI0_CTRL	**Size**	32 bits
Offset	0x44	**Read/Write**	Write only
Register address	0xB0000044	**Initial value**	Unknown

Bit Assignments of R_SCSI0_CTRL

Bit(s)	Name	Description	State/Range
31	id_type	This bit decides whether software selectable SCSI ID or strapped SCSI ID shall be used. If it is set to hardware (0), external hardware straps must be used to determine SCSI ID. If it is set to software (1), the SCSI host controller ID is selected using software. This mode requires the availability of pb4, so pb4 in General Port PB must be configured correctly.	0 = hardware 1 = software
30 - 24	sel_timeout	This field determines the selection timeout interval, expressed in units of 1/300s. The SCSI standard recommends a selection timeout interval of 250ms.	1 - 127
23 - 16	synch_per	This field determines the data period used during synchronous data transfers, expressed in units of 10ns. It also determines the setup/hold time for data. The setup/hold time for data out, as well as the symmetry of the signal on $\overline{\text{s0ack}}$ are generated as half the data period., e.g. 10MHz -> 100ns period -> 50ns setup / 50 ns hold. If the half period is not a multiple of 10ns, the hold time will be longer than the setup time, e.g. 20 MHz -> 50ns period -> 20ns setup / 30ns hold.	2 - 255
15	rst	This bit asserts the RST signal on the SCSI bus. The RST signal must be asserted for at least 25 µs.	0 = no 1 = yes
14	atn	This bit asserts the ATN signal on the SCSI bus. When it is set to yes (1), it tells the target device that the initiator wants to send a message. It is normally only used in manual mode.	0 = no 1 = yes
13	Reserved	-	0
12 - 9	my_id	This field determines the internal initiator id. It is used during arbitration and reselection. The highest bit is only used in wide mode. my_id is ignored if id_type is set to hardware (0).	0 - 15
8	Reserved	-	0
7 - 4	target_id	This field must be set to the id of the target device before starting an arbitration. The highest bit is only used in wide mode.	0 - 15

Bit Assignments of R_SCSI0_CTRL (continued)

Bit(s)	Name	Description	State/Range
3	fast_20	This bit turns off the glitch eater circuitry on the $\overline{\text{s0req}}$ pin. This may be needed when in fast-20 mode, in order to tolerate the worst case duty cycle on $\overline{\text{s0req}}$. Note that it is possible to run both normal and fast-20 mode with both settings. However, in fast-20 mode some pulses on $\overline{\text{s0req}}$ may be lost when this bit is set to no (0), i.e. the glitch eater circuitry is not turned off.	0 = no 1 = yes
2	bus_width	This bit determines the width of the data bus (16/8-bits). It must be set before a data transfer phase is initiated.	0 = narrow 1 = wide
1	synch	This bit selects wether synchronous or asynchronous handshake shall be used in a data transfer phase. It must be set before a data transfer phase is initiated.	0 = asynch 1 = synch
0	enable	This bit enables or disables the SCSI controller. If it is disabled (off) the SCSI controller is reset.	0 = off 1 = on

18.12.2 R_SCSI0_CMD_DATA

SCSI-8 p0 and SCSI-W Command and Data Register, General Characteristics

ID of register	R_SCSI0_CMD_DATA	**Size**	32 bits
Offset	0x40	**Read/Write**	Write only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_SCSI0_CMD_DATA

Bit(s)	Name	Description	State/Range
31 - 27	Reserved	-	0
26	parity_in	This bit enables or disables parity detection on input data.	0 = on 1 = off
25	skip	If this bit is set (on), the bus free phase is entered directly when loosing arbitration.	0 = off 1 = on
24	clr_status	This bit clears the status register and interrupt condition.	0 = nop 1 = yes
23 - 20	asynch_setup	This field determines the setup time used for data during asynchronous handshake, in units of 10ns. SCSI-2 requires at least 55ns and SCSI-3 requires 49ns. Skew in external SCSI-buffers must also be taken into account but 60ns should normally be enough, i.e. asynch_setup = 6.	1 - 15
19 - 16	command	This field is the command to start the SCSI sequencer.	0 = full_din_1 1 = full_dout_1 2 = full_stat_1 3 = resel_din 4 = resel_dout 5 = resel_stat 6 = arb_only 8 = full_din_3 9 = full_dout_3 10 = full_stat_3 11 = man_data_in 12 = man_data_out 13 = man_rat
15 - 0	data_out	This field contains output data used during manual transfers. Upper 8 bits are only used during wide transfers.	0 - 65535

18.12.3 R_SCSI0_DATA

SCSI-8 p0 and SCSI-W Data Register, General Characteristics

ID of register	R_SCSI0_DATA	**Size**	16 bits
Offset	0x40	**Read/Write**	Write only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_SCSI0_DATA

Bit(s)	Name	Description	State/Range
15 - 0	data_out	This field contains output data used during manual transfers. Upper 8 bits are only used during wide transfers.	0 - 65535

Note: This 16-bit register is part of the 32-bit register R_SCSI0_CMD_DATA.

18.12.4 R_SCSI0_CMD

SCSI-8 p0 and SCSI-W Command Register, General Characteristics

ID of register	R_SCSI0_CMD	**Size**	8 bits
Offset	0x42	**Read/Write**	Write only
Register address	0xB0000042	**Initial value**	Unknown

Bit Assignments of R_SCSI0_CMD

Bit(s)	Name	Description	State/Range
7 - 4	asynch_setup	This field determines the setup time used for data during asynchronous handshake, in units of 10ns. SCSI-2 requires at least 55ns and SCSI-3 requires 49ns. Skew in external SCSI-buffers must also be taken into account but 60ns should normally be enough, i.e. asynch_setup = 6.	1 - 15
3 - 0	command	This field is the command to start the SCSI sequencer.	0 = full_din_1 1 = full_dout_1 2 = full_stat_1 3 = resel_din 4 = resel_dout 5 = resel_stat 6 = arb_only 8 = full_din_3 9 = full_dout_3 10 = full_stat_3 11 = man_data_in 12 = man_data_out 13 = man_rat

Note: This 8-bit register is part of the 32-bit register R_SCSI0_CMD_DATA.

18.12.5 R_SCSI0_STATUS_CTRL

SCSI-8 p0 and SCSI-W Status and Control Register, General Characteristics

ID of register	R_SCSI0_STATUS_CTRL	**Size**	8 bits
Offset	0x43	**Read/Write**	Write only
Register address	0xB0000043	**Initial value**	Unknown

Bit Assignments of R_SCSI0_STATUS_CTRL

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2	parity_in	This bit enables or disables parity detection on input data.	0 = on 1 = off
1	skip	If this bit is set (on), the bus free phase is entered directly when loosing arbitration.	0 = off 1 = on
0	clr_status	This bit clears the status register and interrupt condition.	0 = nop 1 = yes

Note: This 8-bit register is part of the 32-bit register R_SCSI0_CMD_DATA.

18.12.6 R_SCSI0_STATUS

SCSI-8 p0 and SCSI-W Status Register, General Characteristics

ID of register	R_SCSI0_STATUS	**Size**	32 bits
Offset	0x48	**Read/Write**	Read only
Register address	0xB0000048	**Initial value**	Unknown

Bit Assignments of R_SCSI0_STATUS

Bit(s)	Name	Description	State/Range
31 - 22	Reserved	-	0
21	parity_error	This is an optional bit to detect parity error during manual mode. It is only valid in manual mode when a REQ has been received. Normally the seq_status field is used to detect parity errors.	0 - 1
20	bus_reset	This bit is set if a reset condition is detected on the bus.	0 = no 1 = yes
19	Reserved	-	0
18 - 15	resel_target	This field contains the SCSI ID of the target that reselected us.	0 - 15
14	resel	This bit is set if a target has reselected us. The targets id is in the resel_target field.	0 = no 1 = yes
13 - 11	curr_phase	This field gives the current SCSI phase. It is only valid in manual mode when a REQ has been received.	0 = ph_undef 1 = ph_resel 2 = ph_command 3 = ph_status 4 = ph_data_out 5 = ph_data_in 6 = ph_msg_out 7 = ph_msg_in

Bit Assignments of R_SCSI0_STATUS (continued)

Bit(s)	Name	Description	State/Range
10 - 6	last_seq_step	This field contains the last step that the sequencer was in, before it was stopped due to some unexpected event.	0 = st_synch_dout 1 = st_synch_din_perr 2 = st_msg_1 3 = st_answer 4 = st_synch_dout_ack 5 = st_synch_din_ack_perr 6 = st_msg_2 7 = st_sdp_disc 8 = st_arbitrate 9 = st_asynch_din 10 = st_manual_req 11 = st_asynch_dout_end 12 = st_synch_din_ack 13 = st_synch_din 14 = st_iwr_good 15 = st_transfer_done 16 = st_wait_free_sdp_disc 17 = st_wait_free_iwr_cc 18 = st_manual_din_prot 20 = st_wait_free_cc 21 = st_wait_free_disc 22 = st_msg_3 23 = st_iwr_cc 24 = st_bus_free 25 = st_asynch_dout 27 = st_iwr 28 = st_manual 29 = st_resel_req 30 = st_transf_cmd 31 = st_cc
5	valid_status	If this bit is set, the seq_status field is valid, which also signals that there is a result from the executing of a command.	0 = no 1 = yes
4 - 0	seq_status	This field contains the result after a command has been executed.	0 = info_seq_complete 1 = info_parity_error 2 = info_unhandled_msg_in 3 = info_unexp_ph_change 4 = info_arb_lost 5 = info_sel_timeout 6 = info_unexp_bf 7 = info_illegal_op 8 = info_rec_recvd 9 = info_reselected 10 = info_unhandled_status 11 = info_bus_reset 12 = info_illegal_bf 13 = info_bus_free

18.12.7 R_SCSI0_DATA_IN

SCSI-8 p0 and SCSI_W Data In Register, General Characteristics

ID of register	R_SCSI0_DATA_IN	**Size**	16 bits
Offset	0x40	**Read/Write**	Read only
Register address	0xB0000040	**Initial value**	Unknown

Bit Assignments of R_SCSI0_DATA_IN

Bit(s)	Name	Description	State/Range
15 - 0	data_in	This field contains data in during manual transfers. Only valid when the seq_status field in R_SCSI0_STATUS is set to info_rec_recvd and no new command has been started. The upper 8 bits are only valid in wide mode.	0 - 65535

18.12.8 R_SCSI1_CTRL

SCSI-8 p1 Control Register, General Characteristics

ID of register	R_SCSI1_CTRL	**Size**	32 bits
Offset	0x54	**Read/Write**	Write only
Register address	0xB0000054	**Initial value**	Unknown

Bit Assignments of R_SCSI1_CTRL

Bit(s)	Name	Description	State/Range
31	id_type	This bit decides whether software selectable SCSI ID or strapped SCSI ID shall be used. If it is set to hardware (0), external hardware straps must be used to determine SCSI ID. If it is set to software (1), the SCSI host controller ID is selected using software. This mode requires the availability of pb7, so pb7 in General Port PB must be configured correctly.	0 = hardware 1 = software
30-24	sel_timeout	This field determines the selection timeout interval, expressed in units of 1/300s. The SCSI standard recommends a selection timeout interval of 250ms.	1 - 127
23-16	synch_per	This field determines the data period used during synchronous data transfers, expressed in units of 10ns. It also determines the setup/hold time for data. The setup/hold time for data out, as well as the symmetry of the signal on $\overline{\text{s1ack}}$ are generated as half the data period., e.g. 10MHz -> 100ns period -> 50ns setup / 50 ns hold. If the half period is not a multiple of 10ns, the hold time will be longer than the setup time, e.g. 20 MHz -> 50ns period -> 20ns setup / 30ns hold.	2 - 255
15	rst	This bit asserts the RST signal on the SCSI bus. The RST signal must be asserted for at least 25 µs.	0 = no 1 = yes
14	atn	This bit asserts the ATN signal on the SCSI bus. When it is set to yes (1), it tells the target device that the initiator wants to send a message. It is normally only used in manual mode.	0 = no 1 = yes
13	Reserved	-	0
12 - 9	my_id	This field determines the internal initiator id. It is used during arbitration and reselection. Bit 12 of this register is not used. my_id is ignored if id_type is set to hardware (0).	0 - 7
8	Reserved	-	0
7 - 4	target_id	This field must be set to the id of the target device before starting an arbitration.Bit 7 of this register is not used.	0 - 7
3	fast_20	This bit turns off the glitch eater circuitry on the $\overline{\text{s1req}}$ pin. This may be needed when in fast-20 mode, in order to tolerate the worst case duty cycle on $\overline{\text{s1req}}$. Note that it is possible to run both normal and fast-20 mode with both settings. However, in fast-20 mode some pulses on $\overline{\text{s0req}}$ may be lost when this bit is set to no (0), i.e. the glitch eater circuitry is not turned off.	0 = no 1 = yes
2	Reserved	-	0

Bit Assignments of R_SCSI1_CTRL Continued

Bit(s)	Name	Description	State/Range
1	synch	This bit selects wether synchronous or asynchronous handshake shall be used in a data transfer phase. It must be set before a data transfer phase is initiated.	0 = asynch 1 = synch
0	enable	This bit enables or disables the SCSI controller. If it is disabled (off) the SCSI controller is reset.	0 = off 1 = on

18.12.9 R_SCSI1_CMD_DATA

SCSI-8 p1 Command and Data Register, General Characteristics

ID of register	R_SCSI1_CMD_DATA	**Size**	32 bits
Offset	0x50	**Read/Write**	Write only
Register address	0xB0000050	**Initial value**	Unknown

Bit Assignments of R_SCSI1_CMD_DATA

Bit(s)	Name	Description	State/Range
3 1 - 27	Reserved	-	0
26	parity_in	This bit enables or disables parity detection on input data.	0 = on 1 = off
25	skip	If this bit is set (on), the bus free phase is entered directly when loosing arbitration.	0 = off 1 = on
24	clr_status	This bit clears the status register and interrupt condition.	0 = nop 1 = yes
23 - 20	asynch_setup	This field determines the setup time used for data during asynchronous handshake, in units of 10ns. SCSI-2 requires at least 55ns and SCSI-3 requires 49ns. Skew in external SCSI-buffers must also be taken into account but 60ns should normally be enough, i.e. asynch_setup = 6.	1 - 15
19 - 16	command	This field is the command to start the SCSI sequencer.	0 = full_din_1 1 = full_dout_1 2 = full_stat_1 3 = resel_din 4 = resel_dout 5 = resel_stat 6 = arb_only 8 = full_din_3 9 = full_dout_3 10 = full_stat_3 11 = man_data_in 12 = man_data_out 13 = man_rat
15 - 0	data_out	This field contains output data used during manual transfers. Bits 15 - 8 of this field are not used by the SCSI interface.	0 - 65535

18.12.10 R_SCSI1_DATA

SCSI-8 p1 Data Register, General Characteristics

ID of register	R_SCSI1_DATA	**Size**	16 bits
Offset	0x50	**Read/Write**	Write only
Register address	0xB0000050	**Initial value**	Unknown

Bit Assignments of R_SCSI1_DATA

Bit(s)	Name	Description	State/Range
15 - 0	data_out	This field contains output data used during manual transfers. Bits 15 - 8 of this field are not used by the SCSI interface.	0 - 65535

Note: This 16-bit register is part of the 32-bit register R_SCSI1_CMD_DATA.

18.12.11 R_SCSI1_CMD

SCSI-8 p1 Command Register, General Characteristics

ID of register	R_SCSI1_CMD	**Size**	8 bits
Offset	0x52	**Read/Write**	Write only
Register address	0xB0000052	**Initial value**	Unknown

Bit Assignments of R_SCSI1_CMD

Bit(s)	Name	Description	State/Range
7 - 4	asynch_setup	This field determines the setup time used for data during asynchronous handshake, in units of 10ns. SCSI-2 requires at least 55ns and SCSI-3 requires 49ns. Skew in external SCSI-buffers must also be taken into account but 60ns should normally be enough, i.e. asynch_setup = 6.	1 - 15
3 - 0	command	This field is the command to start the SCSI sequencer.	0 = full_din_1 1 = full_dout_1 2 = full_stat_1 3 = resel_din 4 = resel_dout 5 = resel_stat 6 = arb_only 8 = full_din_3 9 = full_dout_3 10 = full_stat_3 11 = man_data_in 12 = man_data_out 13 = man_rat

Note: This 8-bit register is part of the 32-bit register R_SCSI1_CMD_DATA.

18.12.12 R_SCSI1_STATUS_CTRL

SCSI-8 p1 Status and Control Register, General Characteristics

ID of register	R_SCSI1_STATUS_CTRL	**Size**	8 bits
Offset	0x53	**Read/Write**	Write only
Register address	0xB0000053	**Initial value**	Unknown

Bit Assignments of R_SCSI1_STATUS_CTRL

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2	parity_in	This bit enables or disables parity detection on input data.	0 = on 1 = off
1	skip	If this bit is set (on), the bus free phase is entered directly when loosing arbitration.	0 = off 1 = on
0	clr_status	This bit clears the status register and interrupt condition.	0 = nop 1 = yes

Note: This 8-bit register is part of the 32-bit register R_SCSI1_CMD_DATA.

18.12.13 R_SCSI1_STATUS

SCSI-8 p1 Status Register, General Characteristics

ID of register	R_SCSI1_STATUS	**Size**	32 bits
Offset	0x58	**Read/Write**	Read only
Register address	0xB0000058	**Initial value**	Unknown

Bit Assignments of R_SCSI1_STATUS

Bit(s)	Name	Description	State/Range
31 - 22	Reserved	-	0
21	parity_error	This is an optional bit to detect parity error during manual mode. It is only valid in manual mode when a REQ has been received. Normally the seq_status field is used to detect parity errors.	0 - 1
20	bus_reset	This bit is set if a reset condition is detected on the bus.	0 = no 1 = yes
19	Reserved	-	0
18 - 15	resel_target	This field contains the SCSI ID of the target that reselected us.	0 - 15
14	resel	This bit is set if a target has reselected us. The targets id is in the resel_target field.	0 = no 1 = yes
13 - 11	curr_phase	This field gives the current SCSI phase. It is only valid in manual mode when a REQ has been received.	0 = ph_undef 1 = ph_resel 2 = ph_command 3 = ph_status 4 = ph_data_out 5 = ph_data_in 6 = ph_msg_out 7 = ph_msg_in

Bit Assignments of R_SCSI1_STATUS (continued)

Bit(s)	Name	Description	State/Range
10 - 6	last_seq_step	This field contains the last step that the sequencer was in, before it was stopped due to some unexpected event.	0 = st_synch_dout 1 = st_synch_din_perr 2 = st_msg_1 3 = st_answer 4 = st_synch_dout_ack 5 = st_synch_din_ack_perr 6 = st_msg_2 7 = st_sdp_disc 8 = st_arbitrate 9 = st_asynch_din 10 = st_manual_req 11 = st_asynch_dout_end 12 = st_synch_din_ack 13 = st_synch_din 14 = st_iwr_good 15 = st_transfer_done 16 = st_wait_free_sdp_disc 17 = st_wait_free_iwr_cc 18 = st_manual_din_prot 20 = st_wait_free_cc 21 = st_wait_free_disc 22 = st_msg_3 23 = st_iwr_cc 24 = st_bus_free 25 = st_asynch_dout 27 = st_iwr 28 = st_manual 29 = st_resel_req 30 = st_transf_cmd 31 = st_cc
5	valid_status	If this bit is set, the seq_status field is valid, which also signals that there is a result from the executing of a command.	0 = no 1 = yes
4 - 0	seq_status	This field contains the result after a command has been executed.	0 = info_seq_complete 1 = info_parity_error 2 = info_unhandled_msg_in 3 = info_unexp_ph_change 4 = info_arb_lost 5 = info_sel_timeout 6 = info_unexp_bf 7 = info_illegal_op 8 = info_rec_recvd 9 = info_reselected 10 = info_unhandled_status 11 = info_bus_reset 12 = info_illegal_bf 13 = info_bus_free

18.12.14 R_SCSI1_DATA_IN

SCSI-8 p1 Data In Register, General Characteristics

ID of register	R_SCSI1_DATA_IN	**Size**	16 bits
Offset	0x50	**Read/Write**	Read only
Register address	0xB0000050	**Initial value**	Unknown

Bit Assignments of R_SCSI1_DATA_IN

Bit(s)	Name	Description	State/Range
15 - 0	data_in	This field contains data in during manual transfers. Only valid when the seq_status field in R_SCSI1_STATUS is set to info_rec_recvd and no new command has been started. Only bits 7 -0 of this field are valid; bits 15 - 8 are ignored.	0 - 65535

18.13 Interrupt Mask and Status Registers

18.13.1 R_IRQ_MASK0_RD

IRQ Mask 0 Read Register, General Characteristics

ID of register	R_IRQ_MASK0_RD	**Size**	32 bits
Offset	0xC0	**Read/Write**	Read only
Register address	0xB00000C0	**Initial value**	Unknown

Bit Assignments of R_IRQ_MASK0_RD

Bit(s)	Name	Description	State/Range
31	nmi_pin	The interrupt from the external $\overline{\text{nmi}}$ pin is read in this field. It is cleared in the external unit connected to the pin. The interrupt has the vector number 0x21.	0 = inactive 1 = active
30	watchdog_nmi	The interrupt from the watchdog timer is read in this field. It is cleared by stopping or restarting the watchdog timer. The interrupt has the vector number 0x21.	0 = inactive 1 = active
29	sqe_test_error	This field contains the individually masked interrupt bit that is set when the sqe_test_error counter attains the value 128. It is cleared by reading the sqe_test_error field of register “R_PHY_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
28	carrier_loss	This field contains the individually masked interrupt bit that is set when the carrier_loss counter attains the value 128. It is cleared by reading the carrier_loss field of register “R_PHY_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
27	deferred	This field contains the individually masked interrupt bit that is set when the deferred counter attains the value 128. It is cleared by reading the deferred field of register “R_TR_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
26	late_col	This field contains the individually masked interrupt bit that is set when the late_col counter attains the value 128. It is cleared by reading the late_col field of register “R_TR_COUNTERS”. The interrupt has the vector number 0x27.	1 = active 0 = inactive
25	multiple_col	This field contains the individually masked interrupt bit that is set when the multiple_col counter attains the value 128. It is cleared by reading the multiple_col field of register “R_TR_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
24	single_col	This field contains the individually masked interrupt bit that is set when the single_col counter attains the value 128. It is cleared by reading the single_col field of register “R_TR_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
23	congestion	This field contains the individually masked interrupt bit that is set when the congestion counter attains the value 128. It is cleared by reading the congestion field of register “R_REC_COUNTERS”. The interrupt has the vector number 0x27. (note 1)	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK0_RD (continued)

Bit(s)	Name	Description	State/Range
22	oversize	This field contains the individually masked interrupt bit that is set when the oversize counter attains the value 128. It is cleared by reading the oversize field of register “R_REC_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
21	alignment_error	This field contains the individually masked interrupt bit that is set when the alignment_error counter attains the value 128. It is cleared by reading the alignment_error field of register “R_REC_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
20	crc_error	This field contains the individually masked interrupt bit that is set when the crc_error counter attains the value 128. It is cleared by reading the crc_error field of register “R_REC_COUNTERS”. The interrupt has the vector number 0x27.	0 = inactive 1 = active
19	overrun	This field contains the individually masked interrupt bit that is set when the network receiver experiences a FIFO overrun condition (congestion error). It is cleared by reading the congestion field of register “R_REC_COUNTERS”. The interrupt has the vector number 0x26. (note 1)	0 = inactive 1 = active
18	underrun	This field contains the individually masked interrupt bit that is set when the network transmitter experiences a FIFO underrun condition. It is cleared by setting the clr_error bit in register “R_NETWORK_TR_CTRL”. The interrupt has the vector number 0x26.	0 = inactive 1 = active
17	excessive_col	This field contains the individually masked interrupt bit that is set when the network transmitter experiences collisions for 16 consecutive transmission attempts. It is set after the first collision if the no_retry field in network interface register “R_NETWORK_TR_CTRL” is set, and when the transmitter stops after the cancel field of “R_NETWORK_TR_CTRL” has been set. The interrupt is cleared by setting the clr_error bit in the “R_NETWORK_TR_CTRL” register. The interrupt has the vector number 0x26.	0 = inactive 1 = active
16	mdio	This field contains the individually masked interrupt bit from the mdio pin. It is generated when the pin is low and should be masked during normal data transfers over the interface. It is cleared in the external unit that is driving the pin. The interrupt has the vector number 0x26.	0 = inactive 1 = active
15	ata_drq3	This field contains the individually masked interrupt bit that is set when a unit on ATA bus 3 requests a DMA transfer. It is cleared at the end of the DMA transfer. The interrupt has the vector number 0x24.	0 = inactive 1 = active
14	ata_drq2	This field contains the individually masked interrupt bit that is set when a unit on ATA bus 2 requests a DMA transfer. It is cleared at the end of the DMA transfer. The interrupt has the vector number 0x24.	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK0_RD (continued)

Bit Assignments of R_IRQ_MASK0_RD (continued)

Bit(s)	Name	Description	State/Range
13	ata_drq1	This field contains the individually masked interrupt bit that is set when a unit on ATA bus 1 requests a DMA transfer. It is cleared at the end of the DMA transfer. The interrupt has the vector number 0x24.	0 = inactive 1 = active
12	ata_drq0	This field contains the individually masked interrupt bit that is set when a unit on ATA bus 0 requests a DMA transfer. It is cleared at the end of the DMA transfer. The interrupt has the vector number 0x24.	0 = inactive 1 = active
11	par0_ecp_cmd ata_irq3	When Parallel Port p0 is in ECP mode, this field contains the individually masked interrupt bit that is set when an ECP command is received in at the port. It is cleared by reading the ecp_cmd field in the "R_PAR0_STATUS_DATA" register. When ATA is in use, this field contains the individually masked interrupt bit that is set when a unit on ATA bus 3 requests an interrupt. It is cleared in the external unit on ATA bus 3. Both of these interrupts have the vector number 0x24. (note 2)	0 = inactive 1 = active
10	par0_peri ata_irq2	When Parallel Port p0 is in use, this field contains the individually masked interrupt bit that is set by the peripheral connected to the port. It is cleared by acknowledging the peri_int bit in register "R_PAR0_CTRL_DATA". When ATA is in use, this field contains the individually masked interrupt bit that is set when a unit on ATA bus 2 requests an interrupt. It is cleared in the external unit on ATA bus 2. These two interrupts both have the vector number 0x24. (note 2)	0 = inactive 1 = active
9	par0_data ata_irq1	When Parallel Port p0 is in use, this field contains the individually masked interrupt bit that is set when input data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared. The interrupt is cleared by reading the data field of register "R_PAR0_STATUS_DATA". When ATA is in use, this field contains the individually masked interrupt bit that is set when a unit on ATA bus 1 requests an interrupt. It is cleared in the external unit on ATA bus 1. These two interrupts both have the vector number 0x24. (note 2)	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK0_RD (continued)

Bit(s)	Name	Description	State/Range
8	par0_ready ata_irq0 mio scsi0	When Parallel Port par0 is in use, this field contains the individually masked interrupt bit that is set when the port is ready to get new data for transmission. The interrupt is cleared by writing new data to the data field of register "R_PAR0_CTRL_DATA" register. This bit should be masked off when the DMA is used for data transfers. When ATA is in use, this field contains the individually masked interrupt bit that is set when a unit on ATA bus 0 requests an interrupt. It is cleared in the external unit on ATA bus 0. This interrupt is detected on the intio pin of the shared RAM interface. It is cleared by setting the i bit of the R_SHARED_RAM_CONFIG register. This interrupt is generated when SCSI controller 0 has finished a command or stopped due to some unexpected event. the interrupt cause can be read in the fields last_seq_step and seq_status in "R_SCSI0_STATUS". It is cleared by setting the clr_status field in "R_SCSI0_CMD_DATA" to yes. These four interrupts all have the vector number 0x24. (note 2)	0 = inactive 1 = active
7	ata_dmaend	This field contains the individually masked interrupt bit that is set when the selected ATA unit releases its DMA request. It should be masked off except when an ATA DMA transfer has been started. The interrupt has the vector number 0x24.	0 = inactive 1 = active
6	Reserved	-	0
5	irq_ext_vector_nr	This field contains the individually masked interrupt bit from the external interrupt pin (irq), when configured for an external vector number. This interrupt is cleared in the external unit connected to the irq pin.	0 = inactive 1 = active
4	irq_int_vector_nr	This field contains the individually masked interrupt bit from the external interrupt pin (irq), when configured for the internally-generated vector number 0x2A. This interrupt is cleared in the external unit connected to the irq pin.	0 = inactive 1 = active
3	ext_dma1	This field contains the individually masked interrupt bit that is set when external DMA channel 1 is stopped. The interrupt should be masked, except when waiting for the completion of a transfer on external DMA channel 1. The interrupt has the internally-generated vector number 0x2D.	0 = inactive 1 = active
2	ext_dma0	This field contains the individually masked interrupt bit that is set when external DMA channel 0 is stopped. The interrupt should be masked, except when waiting for the completion of a transfer on external DMA channel 0. The interrupt has the internally-generated vector number 0x2C.	0 = inactive 1 = active

Bit(s)	Name	Description	State/Range
1	timer1	This field contains the individually masked interrupt bit that is set whenever timer1 reaches its terminal count. The interrupt is cleared by setting the i1 bit in register R_TIMER_CTRL. The interrupt has the internally-generated vector number 0x23.	0 = inactive 1 = active
0	timer0	This field contains the individually masked interrupt bit that is set whenever timer0 reaches its terminal count. The interrupt is cleared by setting the i0 bit in register R_TIMER_CTRL. The interrupt has the internally-generated vector number 0x22.	0 = inactive 1 = active

Note 1: Two similar interrupts are available - overrun and congestion, but usually only one should be enabled. The overrun interrupt should be used if software intervention is necessary when an overrun error occurs. The congestion interrupt should be used if an error count is the only action needed.

Reading the congestion field of R_REC_COUNTERS will clear both the congestion field (bit 23) and the overrun field (bit 19)

Note 2: Bits 11 to 8 are multiplexed between SCSI-8 Port p0, Parallel Port p0, ATA and the shared RAM interface. Register R_GEN_CONFIG is used to select the peripheral device in use.

.

18.13.2 R_IRQ_MASK0_CLR

IRQ Mask 0 Clear Register, General Characteristics

ID of register	R_IRQ_MASK0_CLR	**Size**	32 bits
Offset	0xC0	**Read/Write**	Write only
Register address	0xB00000C0	**Initial value**	Not applicable

Bit Assignments of R_IRQ_MASK0_CLR

Bit(s)	Name	Description	State/Range
31	Reserved	-	0
30	Reserved	-	0
29	sqe_test_error	This field is used to clear the sqe_test_error interrupt mask bit.	0 = nop 1 = clr
28	carrier_loss	This field is used to clear the carrier_loss interrupt mask bit.	0 = nop 1 = clr
27	deferred	This field is used to clear the deferred interrupt mask bit.	0 = nop 1 = clr
26	late_col	This field is used to clear the late_col interrupt mask bit.	0 = nop 1 = clr
25	multiple_col	This field is used to clear the multiple_col interrupt mask bit.	0 = nop 1 = clr
24	single_col	This field is used to clear the single_col interrupt mask bit.	0 = nop 1 = clr
23	congestion	This field is used to clear the congestion interrupt mask bit.	0 = nop 1 = clr
22	oversize	This field is used to clear the oversize interrupt mask bit.	0 = nop 1 = clr
21	alignment_error	This field is used to clear the alignment_error interrupt mask bit.	0 = nop 1 = clr
20	crc_error	This field is used to clear the crc_error interrupt mask bit.	0 = nop 1 = clr
19	overrun	This field is used to clear the overrun interrupt mask bit.	0 = nop 1 = clr
18	underrun	This field is used to clear the underrun interrupt mask bit.	0 = nop 1 = clr
17	excessive_col	This field is used to clear the excessive_col interrupt mask bit.	0 = nop 1 = clr
16	mdio	This field is used to clear the mdio interrupt mask bit.	0 = nop 1 = clr
15	ata_drq3	This field is used to clear the ata_drq3 interrupt mask bit.	0 = nop 1 = clr
14	ata_drq2	This field is used to clear the ata_drq2 interrupt mask bit.	0 = nop 1 = clr
13	ata_drq1	This field is used to clear the ata_drq1 interrupt mask bit.	0 = nop 1 = clr
12	ata_drq0	This field is used to clear the ata_drq0 interrupt mask bit.	0 = nop 1 = clr

Bit Assignments of R_IRQ_MASK0_CLR (continued)

Bit(s)	Name	Description	State/Range
11	par0_ecp_cmd ata_irq3	When Parallel Port p0 is in ECP mode, this field is used to clear the par0_ecp_cmd interrupt mask bit. When ATA is in use, this field is used to clear the ata_irq3 interrupt mask bit.	0 = nop 1 = clr
10	par0_peri ata_irq2	When Parallel Port p0 is in use, this field is used to clear the par0_peri interrupt mask bit. When ATA is in use, this field is used to clear the ata_irq2 interrupt mask bit.	0 = nop 1 = clr
9	par0_data ata_irq1	When Parallel Port p0 is in use, this field is used to clear the par0_data interrupt mask bit. When ATA is in use, this field is used to clear the ata_irq1 interrupt mask bit.	0 = nop 1 = clr
8	par0_ready ata_irq0 mio scsi0	When Parallel Port p0 is in use, this field is used to clear the par0_ready interrupt mask bit. When ATA is in use, this field is used to clear the ata_irq0 interrupt mask bit. When Shared RAM interface is in use, this field is used to clear the mio interrupt mask bit. When SCSI Port 0 is in use, this field is used to clear the scsi0 interrupt mask bit.	0 = nop 1 = clr
7	ata_dmaend	This field is used to clear the ata_dmaend interrupt mask bit.	0 = nop 1 = clr
6	Reserved	-	0
5	irq_ext_vector_nr	This field is used to clear the irq_ext_vector_nr interrupt mask bit.	0 = nop 1 = clr
4	irq_int_vector_nr	This field is used to clear the irq_int_vector_nr interrupt mask bit.	0 = nop 1 = clr
3	ext_dma1	This field is used to clear the ext_dma1 interrupt mask bit.	0 = nop 1 = clr
2	ext_dma0	This field is used to clear the ext_dma0 interrupt mask bit.	0 = nop 1 = clr
1	timer1	This field is used to clear the timer1 interrupt mask bit.	0 = nop 1 = clr
0	timer0	This field is used to clear the timer2 interrupt mask bit.	0 = nop 1 = clr

Note: In this register, only bits written with 1 are cleared. Bits written with 0 are not affected.

18.13.3 R_IRQ_READ0

IRQ Read 0 Register, General Characteristics

ID of register	R_IRQ_READ0	**Size**	32 bits
Offset	0xC4	**Read/Write**	Read only
Register address	0xB00000C4	**Initial value**	Unknown

Bit Assignments of R_IRQ_READ0

Bit(s)	Name	Description	State/Range
31	nmi_pin	This field is used to read the status of the interrupt at the external $\overline{\text{nmi}}$ pin.	0 = inactive 1 = active
30	watchdog_nmi	This field is used to read the status of the interrupt from the watchdog timer.	0 = inactive 1 = active
29	sqe_test_error	This field is used to read the status of the sqe_test_error interrupt prior to the individual mask.	0 = inactive 1 = active
28	carrier_loss	This field is used to read the status of the carrier_loss interrupt prior to the individual mask.	0 = inactive 1 = active
27	deferred	This field is used to read the status of the deferred interrupt prior to the individual mask.	0 = inactive 1 = active
26	late_col	This field is used to read the status of the late_col interrupt prior to the individual mask.	0 = inactive 1 = active
25	multiple_col	This field is used to read the status of the multiple_col interrupt prior to the individual mask.	1 = active 0 = inactive
24	single_col	This field is used to read the status of the single_col interrupt prior to the individual mask.	0 = inactive 1 = active
23	congestion	This field is used to read the status of the congestion interrupt prior to the individual mask.	0 = inactive 1 = active
22	oversize	This field is used to read the status of the oversize interrupt prior to the individual mask.	0 = inactive 1 = active
21	alignment_error	This field is used to read the status of the alignment_error interrupt prior to the individual mask.	0 = inactive 1 = active
20	crc_error	This field is used to read the status of the crc_error interrupt prior to the individual mask.	0 = inactive 1 = active
19	overrun	This field is used to read the status of the overrun interrupt prior to the individual mask.	0 = inactive 1 = active
18	underrun	This field is used to read the status of the underrun interrupt prior to the individual mask.	0 = inactive 1 = active
17	excessive_col	This field is used to read the status of the excessive_col interrupt prior to the individual mask.	0 = inactive 1 = active
16	mdio	This field is used to read the status of the mdio interrupt prior to the individual mask.	1 = active 0 = inactive
15	ata_drq3	This field is used to read the status of the ata_drq3 interrupt prior to the individual mask.	1 = active 0 = inactive
14	ata_drq2	This field is used to read the status of the ata_drq2 interrupt prior to the individual mask.	0 = inactive 1 = active
13	ata_drq1	This field is used to read the status of the ata_drq1 interrupt prior to the individual mask.	0 = inactive 1 = active
12	ata_drq0	This field is used to read the status of the ata_drq0 interrupt prior to the individual mask.	0 = inactive 1 = active

Bit Assignments of R_IRQ_READ0 (continued)

Bit(s)	Name	Description	State/Range
11	par0_ecp_cmd ata_irq3	When Parallel Port p0 is in ECP mode, this field is used to read the status of the par0_ecp_cmd interrupt prior to the individual mask. When ATA is in use, this field is used to read the status of the ata_irq3 interrupt prior to the individual mask.	0 = inactive 1 = active
10	par0_peri ata_irq2	When Parallel Port p0 is in use, this field is used to read the status of the par0_peri interrupt prior to the individual mask. When ATA is in use, this field is used to read the status of the ata_irq2 interrupt prior to the individual mask.	0 = inactive 1 = active
9	par0_data ata_irq1	When Parallel Port p0 is in use, this field is used to read the status of the par0_data interrupt prior to the individual mask. When ATA is in use, this field is used to read the status of the ata_irq1 interrupt prior to the individual mask.	0 = inactive 1 = active
8	par0_ready ata_irq0 mio scsi0	When Parallel Port p0 is in use, this field is used to read the status of the par0_ready interrupt prior to the individual mask. When ATA is in use, this field is used to read the status of the ata_irq0 interrupt prior to the individual mask. This field is used to read the status of the mio interrupt prior to the individual mask. This field is used to read the status of the scsi0 interrupt prior to the individual mask.	0 = inactive 1 = active
7	ata_dmaend	This field is used to read the status of the ata_dmaend interrupt prior to the individual mask.	0 = inactive 1 = active
6	Reserved	-	0
5	irq_ext_vector_nr	This field is used to read the status of the irq_ext_vector_nr interrupt prior to the individual mask.	0 = inactive 1 = active
4	irq_int_vector_nr	This field is used to read the status of the irq_int_vector_nr interrupt prior to the individual mask.	0 = inactive 1 = active
3	ext_dma1	This field is used to read the status of the ext_dma1 interrupt prior to the individual mask.	0 = inactive 1 = active
2	ext_dma0	This field is used to read the status of the ext_dma0 interrupt prior to the individual mask.	0 = inactive 1 = active
1	timer1	This field is used to read the status of the timer1 interrupt prior to the individual mask.	0 = inactive 1 = active
0	timer0	This field is used to read the status of the timer0 interrupt prior to the individual mask.	0 = inactive 1 = active

18.13.4 R_IRQ_MASK0_SET

IRQ Mask 0 Set Register, General Characteristics

ID of register	R_IRQ_MASK0_SET	Size	32 bits
Offset	0xC4	Read/Write	Write only
Register address	0xB00000C4	Initial value	Not applicable

Bit Assignments of R_IRQ_MASK0_SET

Bit(s)	Name	Description	State/Range
31	Reserved	-	0
30	Reserved	-	0
29	sqe_test_error	This field is used to set the individual mask bit for the sqe_test_error interrupt. The interrupt has the internally-generated vector number 0x27.	0 = nop 1 = set
28	carrier_loss	This field is used to set the individual mask bit for the carrier_loss interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
27	deferred	This field is used to set the individual mask bit for the deferred interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
26	late_col	This field is used to set the individual mask bit for the late_col interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
25	multiple_col	This field is used to set the individual mask bit for the multiple_col interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
24	single_col	This field is used to set the individual mask bit for the single_col interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
23	congestion	This field is used to set the individual mask bit for the congestion interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
22	oversize	This field is used to set the individual mask bit for the oversize interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
21	alignment_error	This field is used to set the individual mask bit for the alignment_error interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
20	crc_error	This field is used to set the individual mask bit for the crc_error interrupt. The interrupt has the vector number 0x27.	0 = nop 1 = set
19	overrun	This field is used to set the individual mask bit for the overrun interrupt. The interrupt has the vector number 0x26.	0 = nop 1 = set
18	underrun	This field is used to set the individual mask bit for the underrun interrupt. The interrupt has the vector number 0x26.	0 = nop 1 = set

Bit Assignments of R_IRQ_MASK0_SET (continued)

Bit Assignments of R_IRQ_MASK0_SET (continued)

Bit(s)	Name	Description	State/Range
17	excessive_col	This field is used to set the individual mask bit for the excessive_col interrupt. The interrupt has the vector number 0x26.	0 = nop 1 = set
16	mdio	This field is used to set the individual mask bit for the mdio interrupt. The interrupt has the vector number 0x26.	0 = nop 1 = set
15	ata_drq3	This field is used to set the individual mask bit for the ata_drq3 interrupt. The interrupt has the vector number 0x24.	0 = nop 1 = set
14	ata_drq2	This field is used to set the individual mask bit for the ata_drq2 interrupt. The interrupt has the vector number 0x24.	0 = nop 1 = set
13	ata_drq1	This field is used to set the individual mask bit for the ata_drq1 interrupt. The interrupt has the vector number 0x24.	0 = nop 1 = set
12	ata_drq0	This field is used to set the individual mask bit for the ata_drq0 interrupt. The interrupt has the vector number 0x24.	0 = nop 1 = set
11	par0_ecp_cmd ata_irq3	This field is used to set the individual mask bit for the par0_ecp_cmd interrupt. This field is used to set the individual mask bit for the ata_irq3 interrupt. Both interrupts have the vector number 0x24.	0 = nop 1 = set
10	par0_peri ata_irq2	This field is used to set the individual mask bit for the par0_peri interrupt. This field is used to set the individual mask bit for the ata_irq2 interrupt. Both interrupts have the vector number 0x24.	0 = nop 1 = set
9	par0_data ata_irq1	This field is used to set the individual mask bit for the par0_data interrupt. This field is used to set the individual mask bit for the ata_irq1 interrupt. Both interrupts have the vector number 0x24.	0 = nop 1 = set

Bit(s)	Name	Description	State/Range
8	par0_ready ata_irq0 mio scsi0	This field is used to set the individual mask bit for the par0_ready interrupt. This field is used to set the individual mask bit for the ata_irq0 interrupt. This field is used to set the individual mask bit for the mio interrupt. This field is used to set the individual mask bit for the scsi0 interrupt. These four interrupts all have the vector number 0x24.	0 = nop 1 = set
7	ata_dmaend	This field is used to set the individual mask bit for the ata_dmaend interrupt. The interrupt has the vector number 0x24.	0 = nop 1 = set
6	Reserved	-	0

5	irq_ext_vector_nr	This field is used to set the individual mask bit for the irq_ext_vector_nr interrupt. This interrupt has an external vector number. (note 1)	0 = nop 1 = set
4	irq_int_vector_nr	This field is used to set the individual mask bit for the irq_int_vector_nr interrupt. The interrupt has the internally-generated vector number 0x24. (note 1)	0 = nop 1 = set
3	ext_dma1	This field is used to set the individual mask bit for the ext_dma1 interrupt. The interrupt has the vector number 0x2D.	0 = nop 1 = set
2	ext_dma0	This field is used to set the individual mask bit for the ext_dma0 interrupt. The interrupt has the vector number 0x2C.	0 = nop 1 = set
1	timer1	This field is used to set the individual mask bit for the timer1 interrupt. The interrupt has the vector number 0x23.	0 = nop 1 = set
0	timer0	This field is used to set the individual mask bit for the timer0 interrupt. The interrupt has the vector number 0x22.	0 = nop 1 = set

Note 1: External interrupt with the external vector is enabled if the mask for irq_ext_vector_nr is set and the mask for irq_int_vector_nr is cleared.

Note 2: In this register, only bits written with 1 are set. Bits written with 0 are not affected.

18.13.5 R_IRQ_MASK1_RD

IRQ Mask 1 Read Register, General Characteristics

ID of register	R_IRQ_MASK1_RD	**Size**	32 bits
Offset	0xC8	**Read/Write**	Read only
Register address	0xB00000C8	**Initial value**	Unknown

Bit Assignments of R_IRQ_MASK1_RD

Bit(s)	Name	Description	State/Range
31	sw_int7	This field contains the interrupt bit that is set when field sw_int7 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
30	sw_int6	This field contains the interrupt bit that is set when field sw_int6 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
29	sw_int5	This field contains the interrupt bit that is set when field sw_int5 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
28	sw_int4	This field contains the interrupt bit that is set when field sw_int4 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
27	sw_int3	This field contains the interrupt bit that is set when field sw_int3 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
26	sw_int2	This field contains the interrupt bit that is set when field sw_int2 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
25	sw_int1	This field contains the interrupt bit that is set when field sw_int1 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
24	sw_int0	This field contains the interrupt bit that is set when field sw_int0 in register R_IRQ_MASK1_SET is set. The interrupt has the vector number 0x29.	0 = inactive 1 = active
23-20	Reserved	-	0
19	par1_ecp_cmd	This field contains the individually masked interrupt bit that is set when Parallel Port p1 receives a command in ECP mode. The interrupt is cleared by reading the ecp_cmd field in register R_PAR1_STATUS_DATA. The interrupt has the vector number 0x25.	0 = inactive 1 = active
18	par1_peri	This field contains the individually masked interrupt bit for the par1_peri interrupt that is set by the peripheral connected to Parallel Port p1. The interrupt is cleared by acknowledging the peri_int bit in register R_PAR1_CTRL_DATA. The interrupt has the vector number 0x25.	0 = inactive 1 = active
17	par1_data	When Parallel Port p1 is in use, this field contains the individually masked interrupt bit that is set when input data is available on the port. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared. The interrupt is cleared by reading the data field of register R_PAR1_STATUS_DATA. The interrupt has the vector number 0x25.	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK1_RD (continued)

Bit(s)	Name	Description	State/Range
16	par1_ready scsi1	When Parallel Port p1 is in use, this field contains the individually masked interrupt bit that is set when the port is ready to get new data for transmission. The interrupt is cleared by writing new data to the data field of register R_PAR1_CTRL_DATA. This bit should be masked off when the DMA is used for data transfers. The interrupt has the vector number 0x25. This interrupt is generated when SCSI controller 1 has finished a command or stopped due to some unexpected event. the interrupt cause can be read in the fields last_seq_step and seq_status in R_SCSI1_STATUS. It is cleared by setting the clr_status field in R_SCSI1_CMD_DATA to yes. (note 1)	0 = inactive 1 = active
15	ser3_ready	This field contains the individually masked interrupt bit that is set when Asynchronous or Synchronous Serial Port p3 is ready to get new data for transmission. The interrupt is cleared by writing new data to the data_out field of register R_SERIAL3_CTRL or to the R_SERIAL3_TR_DATA register. This bit should be masked off when the DMA is used for data transfers. The interrupt has the vector number 0x28.	0 = inactive 1 = active
14	ser3_data	This field contains the individually masked interrupt bit that is set when input data is available at Asynchronous or Synchronous Serial Port p3. The interrupt is cleared by reading the data_in field of register R_SERIAL3_READ or the R_SERIAL3_REC_DATA register. The interrupt has the vector number 0x28.	0 = inactive 1 = active
13	ser2_ready	This field contains the individually masked interrupt bit that is set when Asynchronous Serial Port p2 is ready to get new data for transmission. The interrupt is cleared by writing new data to the data_out field of register R_SERIAL2_CTRL or the R_SERIAL2_TR_DATA register. This bit should be masked off when the DMA is used for data transfers. The interrupt has the vector number 0x28.	0 = inactive 1 = active
12	ser2_data	This field contains the individually masked interrupt bit that is set when input data is available at Asynchronous Serial Port p2. The interrupt is cleared by reading the data_in field of register R_SERIAL2_READ or the R_SERIAL2_REC_DATA register. The interrupt has the vector number 0x28.	0 = inactive 1 = active
11	ser1_ready	This field contains the individually masked interrupt bit that is set when Asynchronous or Synchronous Serial Port p1 is ready to get new data for transmission. The interrupt is cleared by writing new data to the data_out field of register R_SERIAL1_CTRL or to the R_SERIAL1_TR_DATA register. This bit should be masked off when the DMA is used for data transfers. The interrupt has the vector number 0x28.	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK1_RD (continued)

Bit Assignments of R_IRQ_MASK1_RD (continued)

Bit(s)	Name	Description	State/Range
10	ser1_data	This field contains the individually masked interrupt bit that is set when input data is available at Asynchronous or Synchronous Serial Port p1. The interrupt is cleared by reading the data_in field of register R_SERIAL1_READ or the R_SERIAL1_REC_DATA register. The interrupt has the vector number 0x28.	0 = inactive 1 = active
9	ser0_ready	This field contains the individually masked interrupt bit that is set when Asynchronous Serial Port p0 is ready to get new data for transmission. The interrupt is cleared by writing new data to the data_out field of register R_SERIAL0_CTRL or to the R_SERIAL0_TR_DATA register. This bit should be masked off when the DMA is used for data transfers. The interrupt has the vector number 0x28.	0 = inactive 1 = active
8	ser0_data	This field contains the individually masked interrupt bit that is set when input data is available at Asynchronous Serial Port p0. The interrupt is cleared by reading the data_in field of register R_SERIAL0_READ or the R_SERIAL0_REC_DATA register. The interrupt has the vector number 0x28.	0 = inactive 1 = active
7	pa7	This field contains the individually masked bit for the interrupt on pin pa7 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa7. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
6	pa6	This field contains the individually masked bit for the interrupt on pin pa6 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa6. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
5	pa5	This field contains the individually masked bit for the interrupt on pin pa5 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa5. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
4	pa4	This field contains the individually masked bit for the interrupt on pin pa4 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa4. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
3	pa3	This field contains the individually masked bit for the interrupt on pin pa3 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa3. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
2	pa2	This field contains the individually masked bit for the interrupt on pin pa2 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa2. The interrupt has the vector number 0x2B.	0 = inactive 1 = active

Bit(s)	Name	Description	State/Range

1	pa1	This field contains the individually masked bit for the interrupt on pin pa1 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa1. The interrupt has the vector number 0x2B.	0 = inactive 1 = active
0	pa0	This field contains the individually masked bit for the interrupt on pin pa0 of General Port PA, when the port is used for interrupt handling. The interrupt is cleared in the external unit connected to pa0. The interrupt has the vector number 0x2B.	0 = inactive 1 = active

Note: Bit 16 is multiplexed between SCSI-8 Port p1 and Parallel Port p1. Register R_GEN_CONFIG is used to select which peripheral device to use.

18.13.6 R_IRQ_MASK1_CLR

IRQ Mask 1 Clear Register, General Characteristics

ID of register	R_IRQ_MASK1_CLR	**Size**	32 bits
Offset	0xC8	**Read/Write**	Write only
Register address	0xB00000C8	**Initial value**	Not applicable

Bit Assignments of R_IRQ_MASK1_CLR

Bit(s)	Name	Description	State/Range
31	sw_int7	This field is used to clear the sw_int7 interrupt mask bit.	0 = nop 1 = clr
30	sw_int6	This field is used to clear the sw_int6 interrupt mask bit.	0 = nop 1 = clr
29	sw_int5	This field is used to clear the sw_int5 interrupt mask bit.	0 = nop 1 = clr
28	sw_int4	This field is used to clear the sw_int4 interrupt mask bit.	0 = nop 1 = clr
27	sw_int3	This field is used to clear the sw_int3 interrupt mask bit.	0 = nop 1 = clr
26	sw_int2	This field is used to clear the sw_int2 interrupt mask bit.	0 = nop 1 = clr
25	sw_int1	This field is used to clear the sw_int1 interrupt mask bit.	0 = nop 1 = clr
24	sw_int0	This field is used to clear the sw_int0 interrupt mask bit.	0 = nop 1 = clr
23-20	Reserved	-	0
19	par1_ecp_cmd	This field is used to clear the par1_ecp_cmd interrupt mask bit.	0 = nop 1 = clr
18	par1_peri	This field is used to clear the par1_peri interrupt mask bit.	0 = nop 1 = clr
17	par1_data	This field is used to clear the par1_data interrupt mask bit.	0 = nop 1 = clr
16	par1_ready scsi1	When Parallel Port p1 is in use, this field is used to clear the par1_ready interrupt mask bit. When SCSI Port 1 is in use, this field is used to clear the scsi1 interrupt mask bit.	0 = nop 1 = clr
15	ser3_ready	This field is used to clear the ser3_ready interrupt mask bit.	0 = nop 1 = clr
14	ser3_data	This field is used to clear the ser3_data interrupt mask bit.	0 = nop 1 = clr
13	ser2_ready	This field is used to clear the ser2_ready interrupt mask bit.	0 = nop 1 = clr
12	ser2_data	This field is used to clear the ser2_data interrupt mask bit.	0 = nop 1 = clr
11	ser1_ready	This field is used to clear the ser1_ready interrupt mask bit.	0 = nop 1 = clr
10	ser1_data	This field is used to clear the ser1_data interrupt mask bit.	0 = nop 1 = clr

Bit Assignments of R_IRQ_MASK1_CLR (continued)

Bit(s)	Name	Description	State/Range
9	ser0_ready	This field is used to clear the ser0_ready interrupt mask bit.	0 = nop 1 = clr
8	ser0_data	This field is used to clear the ser0_data interrupt mask bit.	0 = nop 1 = clr
7	pa7	This field is used to clear the pa7 interrupt mask bit.	0 = nop 1 = clr
6	pa6	This field is used to clear the pa6 interrupt mask bit	0 = nop 1 = clr
5	pa5	This field is used to clear the pa5 interrupt mask bit	0 = nop 1 = clr
4	pa4	This field is used to clear the pa4 interrupt mask bit	0 = nop 1 = clr
3	pa3	This field is used to clear the pa3 interrupt mask bit.	0 = nop 1 = clr
2	pa2	This field is used to clear the pa2 interrupt mask bit.	0 = nop 1 = clr
1	pa1	This field is used to clear the pa1 interrupt mask bit	0 = nop 1 = clr
0	pa0	This field is used to clear the pa0 interrupt mask bit	0 = nop 1 = clr

Note: In this register, only bits written with 1 are cleared. Bits written with 0 are not affected.

18.13.7 R_IRQ_READ1

IRQ Read 1 Register, General Characteristics

ID of register	R_IRQ_READ1	**Size**	32 bits
Offset	0xCC	**Read/Write**	Read only
Register address	0xB00000CC	**Initial value**	Unknown

Bit Assignments of R_IRQ_READ1

Bit(s)	Name	Description	State/Range
31	sw_int7	This field is used to read the status of software-generated interrupt sw_int7.	0 = inactive 1 = active
30	sw_int6	This field is used to read the status of software-generated interrupt sw_int6.	0 = inactive 1 = active
29	sw_int5	This field is used to read the status of software-generated interrupt sw_int5.	0 = inactive 1 = active
28	sw_int4	This field is used to read the status of software-generated interrupt sw_int4.	0 = inactive 1 = active
27	sw_int3	This field is used to read the status of software-generated interrupt sw_int3.	0 = inactive 1 = active
26	sw_int2	This field is used to read the status of software-generated interrupt sw_int2.	0 = inactive 1 = active
25	sw_int1	This field is used to read the status of software-generated interrupt sw_int1.	0 = inactive 1 = active
24	sw_int0	This field is used to read the status of software-generated interrupt sw_int0.	0 = inactive 1 = active
23 - 20	Reserved	-	0
19	par1_ecp_cmd	This field is used to read the status of the par1_ecp_cmd interrupt prior to the individual mask.	0 = inactive 1 = active
18	par1_peri	This field is used to read the status of the par1_peri interrupt prior to the individual mask.	0 = inactive 1 = active
17	par1_data	This field is used to read the status of the par1_data interrupt prior to the individual mask.	0 = inactive 1 = active
16	par1_ready scsi1	This field is used to read the status of the par1_ready interrupt prior to the individual mask. This field is used to read the status of the scsi1 interrupt prior to the individual mask.	0 = inactive 1 = active
15	ser3_ready	This field is used to read the status of the ser3_ready interrupt prior to the individual mask.	0 = inactive 1 = active
14	ser3_data	This field is used to read the status of the ser3_data interrupt prior to the individual mask.	0 = inactive 1 = active
13	ser2_ready	This field is used to read the status of the ser2_ready interrupt prior to the individual mask.	0 = inactive 1 = active
12	ser2_data	This field is used to read the status of the ser2_data interrupt prior to the individual mask.	0 = inactive 1 = active
11	ser1_ready	This field is used to read the status of the ser1_ready interrupt prior to the individual mask.	0 = inactive 1 = active
10	ser1_data	This field is used to read the status of the ser1_data interrupt prior to the individual mask.	0 = inactive 1 = active

Bit Assignments of R_IRQ_READ1 (continued)

Bit(s)	Name	Description	State/Range
9	ser0_ready	This field is used to read the status of the ser0_ready interrupt prior to the individual mask.	0 = inactive 1 = active
8	ser0_data	This field is used to read the status of the ser0_data interrupt prior to the individual mask.	0 = inactive 1 = active
7	pa7	When General Port PA is used for interrupt handling, this field is used to read the status of interrupt pa7 prior to the individual mask.	0 = inactive 1 = active
6	pa6	This field is used to read the status of pin pa6 prior to the individual mask.	0 = inactive 1 = active
5	pa5	This field is used to read the status of pin pa5 prior to the individual mask.	0 = inactive 1 = active
4	pa4	This field is used to read the status of pin pa4 prior to the individual mask.	0 = inactive 1 = active
3	pa3	This field is used to read the status of pin pa3 prior to the individual mask.	0 = inactive 1 = active
2	pa2	This field is used to read the status of pin pa2 prior to the individual mask.	0 = inactive 1 = active
1	pa1	This field is used to read the status of pin pa1 prior to the individual mask.	0 = inactive 1 = active
0	pa0	This field is used to read the status of pin pa0 prior to the individual mask.	0 = inactive 1 = active

18.13.8 R_IRQ_MASK1_SET

IRQ Mask 1 Set Register, General Characteristics

ID of register	R_IRQ_MASK1_SET	**Size**	32 bits
Offset	0xCC	**Read/Write**	Write only
Register address	0xB00000CC	**Initial value**	Not applicable

Bit Assignments of R_IRQ_MASK1_SET

Bit(s)	Name	Description	State/Range
31	sw_int7	This field is used to set the individual mask bit for the sw_int7 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
30	sw_int6	This field is used to set the individual mask bit for the sw_int6 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
29	sw_int5	This field is used to set the individual mask bit for the sw_int5 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
28	sw_int4	This field is used to set the individual mask bit for the sw_int4 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
27	sw_int3	This field is used to set the individual mask bit for the sw_int3 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
26	sw_int2	This field is used to set the individual mask bit for the sw_int2 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
25	sw_int1	This field is used to set the individual mask bit for the sw_int1 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
24	sw_int0	This field is used to set the individual mask bit for the sw_int0 interrupt. The interrupt has the vector number 0x29.	0 = nop 1 = set
23 - 20	Reserved	-	0
19	par1_ecp_cmd	This field is used to set the individual mask bit for the par1_ecp_cmd interrupt. The interrupt has the vector number 0x25.	0 = nop 1 = set
18	par1_peri	This field is used to set the individual mask bit for the par1_peri interrupt. The interrupt has the vector number 0x25.	0 = nop 1 = set
17	par1_data	This field is used to set the individual mask bit for the par1_data interrupt. The interrupt has the vector number 0x25.	0 = nop 1 = set
16	par1_ready scsi1	This field is used to set the individual mask bit for the par1_ready interrupt. The interrupt has the vector number 0x25. This field is used to set the individual mask bit for the scsi1 interrupt. The interrupt has the vector number 0x25.	0 = nop 1 = set

Bit Assignments of R_IRQ_MASK1_SET (continued)

Bit(s)	Name	Description	State/Range
15	ser3_ready	This field is used to set the individual mask bit for the ser3_ready interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
14	ser3_data	This field is used to set the individual mask bit for the ser3_data interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
13	ser2_ready	This field is used to set the individual mask bit for the ser2_ready interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
12	ser2_data	This field is used to set the individual mask bit for the ser2_data interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
11	ser1_ready	This field is used to set the individual mask bit for the ser1_ready interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
10	ser1_data	This field is used to set the individual mask bit for the ser1_data interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
9	ser0_ready	This field is used to set the individual mask bit for the ser0_ready interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
8	ser0_data	This field is used to set the individual mask bit for the ser0_data interrupt. The interrupt has the vector number 0x28.	0 = nop 1 = set
7	pa7	This field is used to set the individual mask bit for the interrupt on pin pa7. The interrupt has the vector number 0x2B.	0 = nop 1 = set
6	pa6	This field is used to set the individual mask bit for the interrupt on pin pa6. The interrupt has the vector number 0x2B.	0 = nop 1 = set
5	pa5	This field is used to set the individual mask bit for the interrupt on pin pa5. The interrupt has the vector number 0x2B.	0 = nop 1 = set
4	pa4	This field is used to set the individual mask bit for the interrupt on pin pa4. The interrupt has the vector number 0x2B.	0 = nop 1 = set
3	pa3	This field is used to set the individual mask bit for the interrupt on pin pa3. The interrupt has the vector number 0x2B.	0 = nop 1 = set
2	pa2	This field is used to set the individual mask bit for the interrupt on pin pa2. The interrupt has the vector number 0x2B.	0 = nop 1 = set
1	pa1	This field is used to set the individual mask bit for the interrupt on pin pa1. The interrupt has the vector number 0x2B.	0 = nop 1 = set
0	pa0	This field is used to set the individual mask bit for the interrupt on pin pa0. The interrupt has the vector number 0x2B.	0 = nop 1 = set

Note: In this register, only bits written with 1 are set. Bits written with 0 are not affected.

18.13.9 R_IRQ_MASK2_RD

IRQ Mask 2 Read Register, General Characteristics

ID of register	R_IRQ_MASK2_RD	**Size**	32 bits
Offset	0xD0	**Read/Write**	Read only
Register address	0xB00000D0	**Initial value**	Unknown

Bit Assignments of R_IRQ_MASK2_RD

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	dma8_sub3_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 8, sub-channel 3. It is cleared by writing to the clr_descr field of R_DMA_CH8_SUB3_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
22	dma8_sub2_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 8, sub-channel 2. It is cleared by writing to the clr_descr field of R_DMA_CH8_SUB2_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
21	dma8_sub1_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 8, sub-channel 1. It is cleared by writing to the clr_descr field of R_DMA_CH8_SUB1_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
20	dma8_sub0_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 8, sub-channel 0. It is cleared by writing to the clr_descr field of R_DMA_CH8_SUB0_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
19	dma9_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 9. It is cleared by writing to the clr_eop field of R_DMA_CH9_CLR_INTR. The interrupt has the number 0x39.	0 = inactive 1 = active
18	dma9_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 9. It is cleared by writing to the clr_descr field of R_DMA_CH9_CLR_INTR. The interrupt has the vector number 0x39.	0 = inactive 1 = active
17	dma8_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 8. It is cleared by writing to the clr_eop field of R_DMA_CH8_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
16	dma8_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 8. It is cleared by writing to the clr_descr field of R_DMA_CH8_CLR_INTR. The interrupt has the vector number 0x38.	0 = inactive 1 = active
15	dma7_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 7. It is cleared by writing to the clr_eop field of R_DMA_CH7_CLR_INTR. The interrupt has the vector number 0x37.	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK2_RD (continued)

Bit(s)	Name	Description	State/Range
14	dma7_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 7. It is cleared by writing to the clr_descr field of R_DMA_CH7_CLR_INTR. The interrupt has the vector number 0x37.	0 = inactive 1 = active
13	dma6_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 6. It is cleared by writing to the clr_eop field of R_DMA_CH6_CLR_INTR. The interrupt has the vector number 0x36.	0 = inactive 1 = active
12	dma6_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 6. It is cleared by writing to the clr_descr field of R_DMA_CH6_CLR_INTR. The interrupt has the vector number 0x36.	0 = inactive 1 = active
11	dma5_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 5. It is cleared by writing to the clr_eop field of R_DMA_CH5_CLR_INTR. The interrupt has the vector number 0x35.	0 = inactive 1 = active
10	dma5_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 5. It is cleared by writing to the clr_descr field of R_DMA_CH5_CLR_INTR. The interrupt has the vector number 0x35.	0 = inactive 1 = active
9	dma4_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 4. It is cleared by writing to the clr_eop field of R_DMA_CH4_CLR_INTR. The interrupt has the vector number 0x34.	0 = inactive 1 = active
8	dma4_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 4. It is cleared by writing to the clr_descr field of R_DMA_CH4_CLR_INTR. The interrupt has the vector number 0x34.	0 = inactive 1 = active
7	dma3_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 3. It is cleared by writing to the clr_eop field of R_DMA_CH3_CLR_INTR. The interrupt has the vector number 0x33.	0 = inactive 1 = active
6	dma3_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 3. It is cleared by writing to the clr_descr field of R_DMA_CH3_CLR_INTR. The interrupt has the vector number 0x33.	0 = inactive 1 = active
5	dma2_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 2. It is cleared by writing to the clr_eop field of R_DMA_CH2_CLR_INTR. The interrupt has the vector number 0x32.	0 = inactive 1 = active
4	dma2_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 2. It is cleared by writing to the clr_descr field of R_DMA_CH2_CLR_INTR. The interrupt has the vector number 0x32.	0 = inactive 1 = active

Bit Assignments of R_IRQ_MASK2_RD (continued)

Bit(s)	Name	Description	State/Range
3	dma1_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 1. It is cleared by writing to the clr_eop field of R_DMA_CH1_CLR_INTR. The interrupt has the vector number 0x31.	0 = inactive 1 = active
2	dma1_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 1. It is cleared by writing to the clr_descr field of R_DMA_CH1_CLR_INTR. The interrupt has the vector number 0x31.	0 = inactive 1 = active
1	dma0_eop	This field contains the individually masked end-of-packet interrupt bit for DMA channel 0. It is cleared by writing to the clr_eop field of R_DMA_CH0_CLR_INTR. The interrupt has the vector number 0x30.	0 = inactive 1 = active
0	dma0_descr	This field contains the individually masked descriptor interrupt bit for DMA channel 0. It is cleared by writing to the clr_descr field of R_DMA_CH0_CLR_INTR. The interrupt has the vector number 0x30.	0 = inactive 1 = active

18.13.10 R_IRQ_MASK2_CLR

IRQ Mask 2 Clear Register, General Characteristics

ID of register	R_IRQ_MASK2_CLR	**Size**	32 bits
Offset	0xD0	**Read/Write**	Write only
Register address	0xB00000D0	**Initial value**	Not applicable

Bit Assignments of R_IRQ_MASK2_CLR

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	dma8_sub3_descr	This field is used to clear the dma8_sub3_descr interrupt mask bit.	0 = nop 1 = clr
22	dma8_sub2_descr	This field is used to clear the dma8_sub2_descr interrupt mask bit.	0 = nop 1 = clr
21	dma8_sub1_descr	This field is used to clear the dma8_sub1_descr interrupt mask bit.	0 = nop 1 = clr
20	dma8_sub0_descr	This field is used to clear the dma8_sub0_descr interrupt mask bit.	0 = nop 1 = clr
19	dma9_eop	This field is used to clear the dma9_eop interrupt mask bit.	0 = nop 1 = clr
18	dma9_descr	This field is used to clear the dma9_descr interrupt mask bit.	0 = nop 1 = clr
17	dma8_eop	This field is used to clear the dma8_eop interrupt mask bit.	0 = nop 1 = clr
16	dma8_descr	This field is used to clear the dma8_descr interrupt mask bit.	0 = nop 1 = clr
15	dma7_eop	This field is used to clear the dma7_eop interrupt mask bit.	0 = nop 1 = clr
14	dma7_descr	This field is used to clear the dma7_descr interrupt mask bit.	0 = nop 1 = clr
13	dma6_eop	This field is used to clear the dma6_eop interrupt mask bit.	0 = nop 1 = clr
12	dma6_descr	This field is used to clear the dma6_descr interrupt mask bit.	0 = nop 1 = clr
11	dma5_eop	This field is used to clear the dma5_eop interrupt mask bit.	0 = nop 1 = clr
10	dma5_descr	This field is used to clear the dma5_descr interrupt mask bit.	0 = nop 1 = clr
9	dma4_eop	This field is used to clear the dma4_eop interrupt mask bit.	0 = nop 1 = clr
8	dma4_descr	This field is used to clear the dma4_descr interrupt mask bit.	0 = nop 1 = clr
7	dma3_eop	This field is used to clear the dma3_eop interrupt mask bit.	0 = nop 1 = clr
6	dma3_descr	This field is used to clear the dma3_descr interrupt mask bit.	0 = nop 1 = clr
5	dma2_eop	This field is used to clear the dma2_eop interrupt mask bit.	0 = nop 1 = clr
4	dma2_descr	This field is used to clear the dma2_descr interrupt mask bit.	0 = nop 1 = clr

Bit Assignments of R_IRQ_MASK2_CLR (continued)

Bit(s)	Name	Description	State/Range
3	dma1_eop	This field is used to clear the dma1_eop interrupt mask bit.	0 = nop 1 = clr
2	dma1_descr	This field is used to clear the dma1_descr interrupt mask bit.	0 = nop 1 = clr
1	dma0_eop	This field is used to clear the dma0_eop interrupt mask bit.	0 = nop 1 = clr
0	dma0_descr	This field is used to clear the dma0_descr interrupt mask bit.	0 = nop 1 = clr

Note: In this register, only bits written with 1 are cleared. Bits written with 0 are not affected.

18.13.11 R_IRQ_READ2

IRQ Read 2 Register, General Characteristics

ID of register	R_IRQ_READ2	**Size**	32 bits
Offset	0xD4	**Read/Write**	Read only
Register address	0xB00000D4	**Initial value**	Unknown

Bit Assignments of R_IRQ_READ2

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	dma8_sub3_descr	This field is used to read the status of the dma8_sub3_descr interrupt prior to the individual mask.	0 = inactive 1 = active
22	dma8_sub2_descr	This field is used to read the status of the dma8_sub2_descr interrupt prior to the individual mask.	0 = inactive 1 = active
21	dma8_sub1_descr	This field is used to read the status of the dma8_sub1_descr interrupt prior to the individual mask.	0 = inactive 1 = active
20	dma8_sub0_descr	This field is used to read the status of the dma8_sub0_descr interrupt prior to the individual mask.	0 = inactive 1 = active
19	dma9_eop	This field is used to read the status of the dma9_eop interrupt prior to the individual mask.	0 = inactive 1 = active
18	dma9_descr	This field is used to read the status of the dma9_descr interrupt prior to the individual mask.	0 = inactive 1 = active
17	dma8_eop	This field is used to read the status of the dma8_eop interrupt prior to the individual mask.	0 = inactive 1 = active
16	dma8_descr	This field is used to read the status of the dma8_descr interrupt prior to the individual mask.	0 = inactive 1 = active
15	dma7_eop	This field is used to read the status of the dma7_eop interrupt prior to the individual mask.	0 = inactive 1 = active
14	dma7_descr	This field is used to read the status of the dma7_descr interrupt prior to the individual mask.	0 = inactive 1 = active
13	dma6_eop	This field is used to read the status of the dma6_eop interrupt prior to the individual mask.	0 = inactive 1 = active
12	dma6_descr	This field is used to read the status of the dma6_descr interrupt prior to the individual mask.	0 = inactive 1 = active
11	dma5_eop	This field is used to read the status of the dma5_eop interrupt prior to the individual mask.	0 = inactive 1 = active
10	dma5_descr	This field is used to read the status of the dma5_descr interrupt prior to the individual mask.	0 = inactive 1 = active
9	dma4_eop	This field is used to read the status of the dma4_eop interrupt prior to the individual mask.	0 = inactive 1 = active
8	dma4_descr	This field is used to read the status of the dma4_descr interrupt prior to the individual mask.	0 = inactive 1 = active

Bit Assignments of R_IRQ_READ2 (continued)

Bit(s)	Name	Description	State/Range
7	dma3_eop	This field is used to read the status of the dma3_eop interrupt prior to the individual mask.	0 = inactive 1 = active
6	dma3_descr	This field is used to read the status of the dma3_descr interrupt prior to the individual mask.	0 = inactive 1 = active
5	dma2_eop	This field is used to read the status of the dma2_eop interrupt prior to the individual mask.	0 = inactive 1 = active
4	dma2_descr	This field is used to read the status of the dma2_descr interrupt prior to the individual mask.	0 = inactive 1 = active
3	dma1_eop	This field is used to read the status of the dma1_eop interrupt prior to the individual mask.	0 = inactive 1 = active
2	dma1_descr	This field is used to read the status of the dma1_descr interrupt prior to the individual mask.	0 = inactive 1 = active
1	dma0_eop	This field is used to read the status of the dma0_eop interrupt prior to the individual mask.	0 = inactive 1 = active
0	dma0_descr	This field is used to read the status of the dma0_descr interrupt prior to the individual mask.	0 = inactive 1 = active

18.13.12 R_IRQ_MASK2_SET

IRQ Mask 2 Set Register, General Characteristics

ID of register	R_IRQ_MASK2_SET	Size	32 bits
Offset	0xD4	Read/Write	Write only
Register address	0xB00000D4	Initial value	Not applicable

Bit Assignments of R_IRQ_MASK2_SET

Bit(s)	Name	Description	State/Range
31-24	Reserved	-	0
23	dma8_sub3_descr	This field is used to set the individual mask bit for the dma8_sub3_descr interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
22	dma8_sub2_descr	This field is used to set the individual mask bit for the dma8_sub2_descr interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
21	dma8_sub1_descr	This field is used to set the individual mask bit for the dma8_sub1_descr interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
20	dma8_sub0_descr	This field is used to set the individual mask bit for the dma8_sub0_descr interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
19	dma9_eop	This field is used to set the individual mask bit for the dma9_eop interrupt. The interrupt has the vector number 0x39.	0 = nop 1 = set
18	dma9_descr	This field is used to set the individual mask bit for the dma9_descr interrupt. The interrupt has the vector number 0x39.	0 = nop 1 = set
17	dma8_eop	This field is used to set the individual mask bit for the dma8_eop interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
16	dma8_descr	This field is used to set the individual mask bit for the dma8_descr interrupt. The interrupt has the vector number 0x38.	0 = nop 1 = set
15	dma7_eop	This field is used to set the individual mask bit for the dma7_eop interrupt. The interrupt has the vector number 0x37.	0 = nop 1 = set
14	dma7_descr	This field is used to set the individual mask bit for the dma7_descr interrupt. The interrupt has the vector number 0x37.	0 = nop 1 = set
13	dma6_eop	This field is used to set the individual mask bit for the dma6_eop interrupt. The interrupt has the vector number 0x36.	0 = nop 1 = set
12	dma6_descr	This field is used to set the individual mask bit for the dma6_descr interrupt. The interrupt has the vector number 0x36.	0 = nop 1 = set
11	dma5_eop	This field is used to set the individual mask bit for the dma5_eop interrupt. The interrupt has the vector number 0x35.	0 = nop 1 = set
10	dma5_descr	This field is used to set the individual mask bit for the dma5_descr interrupt. The interrupt has the vector number 0x35.	0 = nop 1 = set

Bit Assignments of R_IRQ_MASK2_SET (continued)

Bit(s)	Name	Description	State/Range
9	dma4_eop	This field is used to set the individual mask bit for the dma4_eop interrupt. The interrupt has the vector number 0x34.	0 = nop 1 = set
8	dma4_descr	This field is used to set the individual mask bit for the dma4_descr interrupt. The interrupt has the vector number 0x34.	0 = nop 1 = set
7	dma3_eop	This field is used to set the individual mask bit for the dma3_eop interrupt. The interrupt has the vector number 0x33.	0 = nop 1 = set
6	dma3_descr	This field is used to set the individual mask bit for the dma3_descr interrupt. The interrupt has the vector number 0x33.	0 = nop 1 = set
5	dma2_eop	This field is used to set the individual mask bit for the dma2_eop interrupt. The interrupt has the vector number 0x32.	0 = nop 1 = set
4	dma2_descr	This field is used to set the individual mask bit for the dma2_descr interrupt. The interrupt has the vector number 0x32.	0 = nop 1 = set
3	dma1_eop	This field is used to set the individual mask bit for the dma1_eop interrupt. The interrupt has the vector number 0x31.	0 = nop 1 = set
2	dma1_descr	This field is used to set the individual mask bit for the dma1_descr interrupt. The interrupt has the vector number 0x31.	0 = nop 1 = set
1	dma0_eop	This field is used to set the individual mask bit for the dma0_eop interrupt. The interrupt has the vector number 0x30.	0 = nop 1 = set
0	dma0_descr	This field is used to set the individual mask bit for the dma0_descr interrupt. The interrupt has the vector number 0x30.	0 = nop 1 = set

Note: In this register, only bits written with 1 are set. Bits written with 0 are not affected.

18.13.13 R_VECT_MASK_RD

Vector Mask Read Register, General Characteristics

ID of register	R_VECT_MASK_RD	Size	32 bits
Offset	0xD8	Read/Write	Read only
Register address	0xB00000D8	Initial value	Unknown

Bit Assignments of R_VECT_MASK_RD

Bit(s)	Name	Description	State/Range
31	usb	This field contains the composed interrupt bit for the USB after the vector mask. The vector number is 0x3F.	0 = inactive 1 = active
30 - 26	Reserved	-	0
25	dma9	This field contains the composed interrupt bit for DMA channel 9 after the vector mask. The vector number is 0x39.	0 = inactive 1 = active
24	dma8	This field contains the composed interrupt bit for DMA channel 8 after the vector mask. The vector number is 0x38.	0 = inactive 1 = active
23	dma7	This field contains the composed interrupt bit for DMA channel 7 after the vector mask. The vector number is 0x37.	0 = inactive 1 = active
22	dma6	This field contains the composed interrupt bit for DMA channel 6 after the vector mask. The vector number is 0x36.	0 = inactive 1 = active
21	dma5	This field contains the composed interrupt bit for DMA channel 5 after the vector mask. The vector number is 0x35.	0 = inactive 1 = active
20	dma4	This field contains the composed interrupt bit for DMA channel 4 after the vector mask. The vector number is 0x34.	0 = inactive 1 = active
19	dma3	This field contains the composed interrupt bit for DMA channel 3 after the vector mask. The vector number is 0x33.	0 = inactive 1 = active
18	dma2	This field contains the composed interrupt bit for DMA channel 2 after the vector mask. The vector number is 0x32.	0 = inactive 1 = active
17	dma1	This field contains the composed interrupt bit for DMA channel 1 after the vector mask. The vector number is 0x31.	0 = inactive 1 = active
16	dma0	This field contains the composed interrupt bit for DMA channel 0 after the vector mask. The vector number is 0x30.	0 = inactive 1 = active
15 - 14	Reserved	-	0
13	ext_dma1	This field contains the composed interrupt bit for external DMA channel 1 after the vector mask. The vector number is 0x2D.	0 = inactive 1 = active
12	ext_dma0	This field contains the composed interrupt bit for external DMA channel 0 after the vector mask. The vector number is 0x2C.	0 = inactive 1 = active
11	pa	This field contains the composed interrupt bit for General Port PA after the vector mask. The vector number is 0x2B.	0 = inactive 1 = active

Bit Assignments of R_VECT_MASK_RD (continued)

Bit(s)	Name	Description	State/Range
10	irq_intnr	This field contains the composed interrupt bit for the $\overline{\text{irq}}$ pin after the vector mask. The vector number is 0x2A.	0 = inactive 1 = active
9	sw	This field contains the composed interrupt bit for the software generated interrupts after the vector mask. The vector number is 0x29.	0 = inactive 1 = active
8	serial	This field contains the composed interrupt bit for the asynchronous and synchronous serial ports after the vector mask. The vector number is 0x28.	0 = inactive 1 = active
7	snmp	This field contains the composed interrupt bit for Ethernet error and statistics counters after the vector mask. The vector number is 0x27.	0 = inactive 1 = active
6	network	This field contains the composed interrupt bit for the network interface after the vector mask. The vector number is 0x26.	0 = inactive 1 = active
5	scsi1 par1	This field contains the composed interrupt bit for SCSI-8 Port p1 or Parallel Port p1 after the vector mask. The vector number is 0x25.	0 = inactive 1 = active
4	scsi0 par0 ata mio	This field contains the composed interrupt bit for SCSI-8 Port p0, SCSI-W Port, Parallel Port p0, the ATA Port or the Shared RAM Port after the vector mask. The vector number is 0x24.	0 = inactive 1 = active
3	timer1	This field contains the composed interrupt bit for timer 1 after the vector mask. The vector number is 0x23.	0 = inactive 1 = active
2	timer0	This field contains the composed interrupt bit for timer 0 after the vector mask. The vector number is 0x22.	0 = inactive 1 = active
1	nmi	This field contains the composed interrupt bit for the NMI. The vector number is 0x21.	0 = inactive 1 = active
0	some	This bit is set if any of the interrupts (except NMI but including $\overline{\text{irq}}$ with an external vector number), are active after the individual and vector masks.	0 = inactive 1 = active

18.13.14 R_VECT_MASK_CLR

Vector Mask Clear Register, General Characteristics

ID of register	R_VECT_MASK_CLR	**Size**	32 bits
Offset	0xD8	**Read/Write**	Write only
Register address	0xB00000D8	**Initial value**	Not applicable

Bit Assignments of R_VECT_MASK_CLR

Bit(s)	Name	Description	State/Range
31	usb	This field clears the vector mask bit for the USB, vector number 0x3F.	0 = nop 1 = clr
30 - 26	Reserved	-	0
25	dma9	This field clears the vector mask bit for DMA channel 9, vector number 0x39.	0 = nop 1 = clr
24	dma8	This field clears the vector mask bit for DMA channel 8, vector number 0x38.	0 = nop 1 = clr
23	dma7	This field clears the vector mask bit for DMA channel 7, vector number 0x37.	0 = nop 1 = clr
22	dma6	This field clears the vector mask bit for DMA channel 6, vector number 0x36.	0 = nop 1 = clr
21	dma5	This field clears the vector mask bit for DMA channel 5, vector number 0x35.	0 = nop 1 = clr
20	dma4	This field clears the vector mask bit for DMA channel 4, vector number 0x34.	0 = nop 1 = clr
19	dma3	This field clears the vector mask bit for DMA channel 3, vector number 0x33.	0 = nop 1 = clr
18	dma2	This field clears the vector mask bit for DMA channel 2, vector number 0x32.	0 = nop 1 = clr
17	dma1	This field clears the vector mask bit for DMA channel 1, vector number 0x31.	0 = nop 1 = clr
16	dma0	This field clears the vector mask bit for DMA channel 0, vector number 0x30.	0 = nop 1 = clr
15 - 14	Reserved	-	0
13	ext_dma1	This field clears the vector mask bit for external DMA channel 1, vector number 0x2D.	0 = nop 1 = clr
12	ext_dma0	This field clears the vector mask bit for external DMA channel 0, vector number 0x2C.	0 = nop 1 = clr
11	pa	This field clears the vector mask bit for General Port PA, vector number 0x2B.	0 = nop 1 = clr
10	irq_intnr	This field clears the vector mask bit for the $\overline{\text{irq}}$ pin, vector number 0x2A.	0 = nop 1 = clr
9	sw	This field clears the vector mask bit for the software generated interrupts, vector number 0x29.	0 = nop 1 = clr
8	serial	This field clears the vector mask bit for the asynchronous serial ports, vector number 0x28.	0 = nop 1 = clr
7	snmp	This field clears the vector mask bit for Ethernet errors and statistics counters, vector number 0x27.	0 = nop 1 = clr
6	network	This field clears the vector mask bit for the network interface, vector number 0x26.	0 = nop 1 = clr

Bit Assignments of R_VECT_MASK_CLR (continued)

Bit(s)	Name	Description	State/Range
5	scsi1 par1	This field clears the vector mask bit for SCSI-8 Port p1 or Parallel Port p1, vector number 0x25.	0 = nop 1 = clr
4	scsi0 par0 ata mio	This field clears the vector mask bit for SCSI-8 Port p0, SCSI-W Port, Parallel Port p0, the ATA Port or the shared RAM Port. Vector number 0x24.	0 = nop 1 = clr
3	timer1	This field clears the vector mask bit for timer 1, vector number 0x23.	0 = nop 1 = clr
2	timer0	This field clears the vector mask bit for timer 0, vector number 0x22.	0 = nop 1 = clr
1 - 0	Reserved	-	0

Note: In this register, only bits written with 1 are cleared. Bits written with 0 are not affected.

18.13.15 R_VECT_READ

Vector Read Register, General Characteristics

ID of register	R_VECT_READ	Size	32 bits
Offset	0xDC	Read/Write	Read only
Register address	0xB00000DC	Initial value	Unknown

Bit Assignments of R_VECT_READ

Bit(s)	Name	Description	State/Range
31	usb	This field is used to read the status of the composed interrupt bit for the USB prior to the vector mask but after the individual mask. Vector number 0x3F.	0 = inactive 1 = active
30 - 26	Reserved	-	0
25	dma9	This field is used to read the status of the composed interrupt bit for DMA channel 9 prior to the vector mask but after the individual mask. Vector number 0x39.	0 = inactive 1 = active
24	dma8	This field is used to read the status of the composed interrupt bit for DMA channel 8 prior to the vector mask but after the individual mask. Vector number 0x38.	0 = inactive 1 = active
23	dma7	This field is used to read the status of the composed interrupt bit for DMA channel 7 prior to the vector mask but after the individual mask. Vector number 0x37.	0 = inactive 1 = active
22	dma6	This field is used to read the status of the composed interrupt bit for DMA channel 6 prior to the vector mask but after the individual mask. Vector number 0x36.	0 = inactive 1 = active
21	dma5	This field is used to read the status of the composed interrupt bit for DMA channel 5 prior to the vector mask but after the individual mask. Vector number 0x35.	0 = inactive 1 = active
20	dma4	This field is used to read the status of the composed interrupt bit for DMA channel 4 prior to the vector mask but after the individual mask. Vector number 0x34.	0 = inactive 1 = active
19	dma3	This field is used to read the status of the composed interrupt bit for DMA channel 3 prior to the vector mask but after the individual mask. Vector number 0x33.	0 = inactive 1 = active
18	dma2	This field is used to read the status of the composed interrupt bit for DMA channel 2 prior to the vector mask but after the individual mask. Vector number 0x32.	0 = inactive 1 = active
17	dma1	This field is used to read the status of the composed interrupt bit for DMA channel 1 prior to the vector mask but after the individual mask. Vector number 0x31.	0 = inactive 1 = active
16	dma0	This field is used to read the status of the composed interrupt bit for DMA channel 0 prior to the vector mask but after the individual mask. Vector number 0x30.	0 = inactive 1 = active
15 - 14	Reserved	-	0
13	ext_dma1	This field is used to read the status of the composed interrupt bit for external DMA channel 1 prior to the vector mask but after the individual mask. Vector number 0x2D.	0 = inactive 1 = active
12	ext_dma0	This field is used to read the status of the composed interrupt bit for external DMA channel 0 prior to the vector mask but after the individual mask. Vector number 0x2C.	0 = inactive 1 = active

Bit Assignments of R_VECT_READ (continued)

Bit(s)	Name	Description	State/Range
11	pa	This field is used to read the status of the composed interrupt bit for General Port PA prior to the vector mask but after the individual mask. Vector number 0x2B.	0 = inactive 1 = active
10	irq_intnr	This field is used to read the status of the composed interrupt bit for the $\overline{\text{irq}}$ pin prior to the vector mask but after the individual mask. Vector number 0x2A.	0 = inactive 1 = active
9	sw	This field is used to read the status of the composed interrupt bit for the software generated interrupts prior to the vector mask but after the individual mask. Vector number 0x29.	0 = inactive 1 = active
8	serial	This field is used to read the status of the composed interrupt bit for the asynchronous serial ports prior to the vector mask but after the individual mask. Vector number 0x28.	0 = inactive 1 = active
7	snmp	This field is used to read the status of the composed interrupt bit for Ethernet error and statistics counters prior to the vector mask but after the individual mask. Vector number 0x27.	0 = inactive 1 = active
6	network	This field is used to read the status of the composed interrupt bit for the network interface prior to the vector mask but after the individual mask. Vector number 0x26.	0 = inactive 1 = active
5	scsi1 par1	This field is used to read the status of the composed interrupt bit for SCSI-8 Port p1 or Parallel Port p1 prior to the vector mask but after the individual mask. Vector number 0x25.	0 = inactive 1 = active
4	scsi0 par0 ata mio	This field is used to read the status of the composed interrupt bit for SCSI-8 Port p0, SCSI-W Port, Parallel Port p0, the ATA Port or the shared RAM Port prior to the vector mask but after the individual mask. Vector number 0x24.	0 = inactive 1 = active
3	timer1	This field is used to read the status of the composed interrupt bit for timer 1 prior to the vector mask but after the individual mask. Vector number 0x23.	0 = inactive 1 = active
2	timer0	This field is used to read the composed interrupt bit for timer 0 prior to the vector mask but after the individual mask. Vector number 0x22.	0 = inactive 1 = active
1	nmi	This field is used to read the status of the composed interrupt bit for the NMI. Vector number 0x21.	0 = inactive 1 = active
0	some	This field is used to read the status of the composed interrupt bit for any of the interrupts (except NMI but including $\overline{\text{irq}}$ with an external vector number), that are active after the individual and vector masks.	0 = inactive 1 = active

18.13.16 R_VECT_MASK_SET

Vector Mask Set Register, General Characteristics

ID of register	R_VECT_MASK_SET	**Size**	32 bits
Offset	0xDC	**Read/Write**	Write only
Register address	0xB00000DC	**Initial value**	Not applicable

Bit Assignments of R_VECT_MASK_SET

Bit(s)	Name	Description	State/Range
31	usb	This field is used to set the vector mask bit for the USB interrupt. Vector number 0x3F.	0 = nop 1 = set
30 - 26	Reserved	-	0
25	dma9	This field is used to set the vector mask bit for the DMA channel 9 interrupt. Vector number 0x39.	0 = nop 1 = set
24	dma8	This field is used to set the vector mask bit for the DMA channel 8 interrupt. Vector number 0x38.	0 = nop 1 = set
23	dma7	This field is used to set the vector mask bit for the DMA channel 7 interrupt. Vector number 0x37.	0 = nop 1 = set
22	dma6	This field is used to set the vector mask bit for the DMA channel 6 interrupt. Vector number 0x36.	0 = nop 1 = set
21	dma5	This field is used to set the vector mask bit for the DMA channel 5 interrupt. Vector number 0x35.	0 = nop 1 = set
20	dma4	This field is used to set the vector mask bit for the DMA channel 4 interrupt. Vector number 0x34.	0 = nop 1 = set
19	dma3	This field is used to set the vector mask bit for the DMA channel 3 interrupt. Vector number 0x33.	0 = nop 1 = set
18	dma2	This field is used to set the vector mask bit for the DMA channel 2 interrupt. Vector number 0x32.	0 = nop 1 = set
17	dma1	This field is used to set the vector mask bit for the DMA channel 1 interrupt. Vector number 0x31.	0 = nop 1 = set
16	dma0	This field is used to set the vector mask bit for the DMA channel 0 interrupt. Vector number 0x30.	0 = nop 1 = set
15 - 14	Reserved	-	0
13	ext_dma1	This field is used to set the vector mask bit for the external DMA channel 1 interrupt. Vector number 0x2D.	0 = nop 1 = set
12	ext_dma0	This field is used to set the vector mask bit for the external DMA channel 0 interrupt. Vector number 0x2C.	0 = nop 1 = set
11	pa	This field is used to set the vector mask bit for the General Port PA interrupt. Vector number 0x2B.	0 = nop 1 = set
10	irq_intnr	This field is used to set the vector mask bit for the $\overline{\text{irq}}$ pin interrupt. Vector number 0x2A.	0 = nop 1 = set
9	sw	This field is used to set the vector mask bit for the software generated interrupt. Vector number 0x29.	0 = nop 1 = set
8	serial	This field is used to set the vector mask bit for the asynchronous serial ports interrupt. Vector number 0x28.	0 = nop 1 = set
7	snmp	This field is used to set the vector mask bit for the Ethernet errors and statistics counters interrupt. Vector number 0x27.	0 = nop 1 = set
6	network	This field is used to set the vector mask bit for the network interface interrupt. Vector number 0x26.	0 = nop 1 = set

Bit Assignments of R_VECT_MASK_SET (continued)

Bit(s)	Name	Description	State/Range
5	scsi1 par1	This field is used to set the vector mask bit for the SCSI-8 Port p1 or Parallel Port p1 interrupt. Vector number 0x25.	0 = nop 1 = set
4	scsi0 par0 ata mio	This field is used to set the vector mask bit for the SCSI-8 Port p0, SCSI-W Port, Parallel Port p0, ATA Port or Shared RAM Port interrupt. Vector number 0x24.	0 = nop 1 = set
3	timer1	This field is used to set the vector mask bit for the timer 1 interrupt. Vector number 0x23.	0 = nop 1 = set
2	timer0	This field is used to set the vector mask bit for the timer 0 interrupt. Vector number 0x22.	0 = nop 1 = set
1	Reserved	-	0
0	Reserved	-	0

Note: In this register, only bits written with 1 are set. Bits written with 0 are not affected.

18.14 DMA Registers

18.14.1 R_SET_EOP

Set End-of-Packet Register, General Characteristics

ID of register	R_SET_EOP	**Size**	32 bits
Offset	0x3C	**Read/Write**	Write only
Register address	0xB000003C	**Initial value**	Not applicable

Bit Assignments of R_SET_EOP

Bit(s)	Name	Description	State/Range
31 - 4	Reserved	-	
3	ch9_eop	Setting this bit to set (1) forces an EOP for DMA channel 9.	0=nop 1=set
2	ch7_eop	Setting this bit to set (1) forces an EOP for DMA channel 7.	0=nop 1=set
1	ch5_eop	Setting this bit to set (1) forces an EOP for DMA channel 5.	0=nop 1=set
0	ch3_eop	Setting this bit to set (1) forces an EOP for DMA channel 3.	0=nop 1=set

Note: Fields set to set (1) force an EOP in the DMA channel, the field values are not saved. Fields set to 0 are not affected.

18.14.2 R_DMA_CH0_HWSW

DMA Channel 0 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH0_HWSW	**Size**	32 bits
Offset	0x100	**Read/Write**	Read/Write
Register address	0xB0000100	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.3 R_DMA_CH0_DESCR

DMA Channel 0 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH0_DESCR	**Size**	32 bits
Offset	0x10C	**Read/Write**	Read/Write
Register address	0xB000010C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.4 R_DMA_CH0_NEXT

DMA Channel 0 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH0_NEXT	**Size**	32 bits
Offset	0x104	**Read/Write**	Read/Write
Register address	0xB0000104	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.5 R_DMA_CH0_BUF

DMA Channel 0 Buffer Register, General Characteristics

ID of register	R_DMA_CH0_BUF	**Size**	32 bits
Offset	0x108	**Read/Write**	Read/Write
Register address	0xB0000108	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.6 R_DMA_CH0_FIRST

DMA Channel 0 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH0_FIRST	**Size**	32 bits
Offset	0x1A0	**Read/Write**	Read/Write
Register address	0xB00001A0	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.7 R_DMA_CH0_CMD

DMA Channel 0 Command Register, General Characteristics

ID of register	R_DMA_CH0_CMD	**Size**	8 bits
Offset	0x1D0	**Read/Write**	Read/Write
Register address	0xB00001D0	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH0_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: This command holds the DMA channel in its current state. start: This command tells the DMA channel to start processing the list at R_DMA_CH0_FIRST. restart: Restart tells the DMA channel to restart after end-of-list has been reached. continue: This command tells the DMA channel to continue after a successful hold command. reset: This command resets the DMA channel and its FIFOs.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.8 R_DMA_CH0_CLR_INTR

DMA Channel 0 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH0_CLR_INTR	**Size**	8 bits
Offset	0x1D1	**Read/Write**	Write only
Register address	0xB00001D1	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH0_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.9 R_DMA_CH0_STATUS

DMA Channel 0 Status Register, General Characteristics

ID of register	R_DMA_CH0_STATUS	**Size**	8 bits
Offset	0x1D2	**Read/Write**	Read only
Register address	0xB00001D2	**Initial value**	Unknown

Bit Assignments of R_DMA_CH0_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 0 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.10 R_DMA_CH1_HWSW

DMA Channel 1 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH1_HWSW	**Size**	32 bits
Offset	0x110	**Read/Write**	Read/Write
Register address	0xB0000110	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.11 R_DMA_CH1_DESCR

DMA Channel 1 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH1_DESCR	**Size**	32 bits
Offset	0x11C	**Read/Write**	Read/Write
Register address	0xB000011C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor of the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.12 R_DMA_CH1_NEXT

DMA Channel 1 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH1_NEXT	**Size**	32 bits
Offset	0x114	**Read/Write**	Read/Write
Register address	0xB0000114	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.13 R_DMA_CH1_BUF

DMA Channel 1 Buffer Register, General Characteristics

ID of register	R_DMA_CH1_BUF	**Size**	32 bits
Offset	0x118	**Read/Write**	Read/Write
Register address	0xB0000118	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.14 R_DMA_CH1_FIRST

DMA Channel 1 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH1_FIRST	**Size**	32 bits
Offset	0x1A4	**Read/Write**	Read/Write
Register address	0xB00001A4	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_FIRST

Bit(s)	**Name**	**Description**	**State/Range**
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.15 R_DMA_CH1_CMD

DMA Channel 1 Command Register, General Characteristics

ID of register	R_DMA_CH1_CMD	**Size**	8 bits
Offset	0x1D4	**Read/Write**	Read/Write
Register address	0xB00001D4	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH1_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH1_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH1_DESCR. start: Start processing list at R_DMA_CH1_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH1_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.16 R_DMA_CH1_CLR_INTR

DMA Channel 1 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH1_CLR_INTR	**Size**	8 bits
Offset	0x1D5	**Read/Write**	Write only
Register address	0xB00001D5	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH1_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.17 R_DMA_CH1_STATUS

DMA Channel 1 Status Register, General Characteristics

ID of register	R_DMA_CH1_STATUS	**Size**	8 bits
Offset	0x1D6	**Read/Write**	Read only
Register address	0xB00001D6	**Initial value**	Unknown

Bit Assignments of R_DMA_CH1_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 1 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.18 R_DMA_CH2_HWSW

DMA Channel 2 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH2_HWSW	**Size**	32 bits
Offset	0x120	**Read/Write**	Read/Write
Register address	0xB0000120	**Initial value**	Unknown

Bit Assignments of R_DMA_CH2_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.19 R_DMA_CH2_DESCR

DMA Channel 2 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH2_DESCR	**Size**	32 bits
Offset	0x12C	**Read/Write**	Read/Write
Register address	0xB000012C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH2_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.20 R_DMA_CH2_NEXT

DMA Channel 2 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH2_NEXT	**Size**	32 bits
Offset	0x124	**Read/Write**	Read/Write
Register address	0xB0000124	**Initial value**	Unknown

Bit Assignments of R_DMA_CH2_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.21 R_DMA_CH2_BUF

DMA Channel 2 Buffer Register, General Characteristics

ID of register	R_DMA_CH2_BUF	**Size**	32 bits
Offset	0x128	**Read/Write**	Read/Write
Register address	0xB0000128	**Initial value**	Unknown

Bit Assignments of R_DMA_CH2_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.22 R_DMA_CH2_FIRST

DMA Channel 2 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH2_FIRST	Size	32 bits
Offset	0x1A8	Read/Write	Read/Write
Register address	0xB00001A8	Initial value	Unknown

Bit Assignments of R_DMA_CH2_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.23 R_DMA_CH2_CMD

DMA Channel 2 Command Register, General Characteristics

ID of register	R_DMA_CH2_CMD	**Size**	8 bits
Offset	0x1D8	**Read/Write**	Read/Write
Register address	0xB00001D8	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH2_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH2_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH2_DESCR. start: Start processing list at R_DMA_CH2_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH2_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.24 R_DMA_CH2_CLR_INTR

DMA Channel 2 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH2_CLR_INTR	**Size**	8 bits
Offset	0x1D9	**Read/Write**	Write only
Register address	0xB00001D9	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH2_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.25 R_DMA_CH2_STATUS

DMA Channel 2 Status Register, General Characteristics

ID of register	R_DMA_CH2_STATUS	**Size**	8 bits
Offset	0x1DA	**Read/Write**	Read only
Register address	0xB00001DA	**Initial value**	Unknown

Bit Assignments of R_DMA_CH2_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 2 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.26 R_DMA_CH3_HWSW

DMA Channel 3 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH3_HWSW	**Size**	32 bits
Offset	0x130	**Read/Write**	Read/Write
Register address	0xB0000130	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.27 R_DMA_CH3_DESCR

DMA Channel 3 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH3_DESCR	**Size**	32 bits
Offset	0x13C	**Read/Write**	Read/Write
Register address	0xB000013C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.28 R_DMA_CH3_NEXT

DMA Channel 3 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH3_NEXT	**Size**	32 bits
Offset	0x134	**Read/Write**	Read/Write
Register address	0xB0000134	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.29 R_DMA_CH3_BUF

DMA Channel 3 Buffer Register, General Characteristics

ID of register	R_DMA_CH3_BUF	**Size**	32 bits
Offset	0x138	**Read/Write**	Read/Write
Register address	0xB0000138	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.30 R_DMA_CH3_FIRST

DMA Channel 3 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH3_FIRST	**Size**	32 bits
Offset	0x1AC	**Read/Write**	Read/Write
Register address	0xB00001AC	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.31 R_DMA_CH3_CMD

DMA Channel 3 Command Register, General Characteristics

ID of register	R_DMA_CH3_CMD	Size	8 bits
Offset	0x1DC	Read/Write	Read/Write
Register address	0xB00001DC	Initial value	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH3_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH3_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH3_DESCR. start: Start processing list at R_DMA_CH3_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH3_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.32 R_DMA_CH3_CLR_INTR

DMA Channel 3 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH3_CLR_INTR	**Size**	8 bits
Offset	0x1DD	**Read/Write**	Write only
Register address	0xB00001DD	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH3_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.33 R_DMA_CH3_STATUS

DMA Channel 3 Status Register, General Characteristics

ID of register	R_DMA_CH3_STATUS	**Size**	8 bits
Offset	0x1DE	**Read/Write**	Read only
Register address	0xB00001DE	**Initial value**	Unknown

Bit Assignments of R_DMA_CH3_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 3 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.34 R_DMA_CH4_HWSW

DMA Channel 4 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH4_HWSW	Size	32 bits
Offset	0x140	Read/Write	Read/Write
Register address	0xB0000140	Initial value	Unknown

Bit Assignments of R_DMA_CH4_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.35 R_DMA_CH4_DESCR

DMA Channel 4 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH4_DESCR	Size	32 bits
Offset	0x14C	Read/Write	Read/Write
Register address	0xB000014C	Initial value	Unknown

Bit Assignments of R_DMA_CH4_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.36 R_DMA_CH4_NEXT

DMA Channel 4 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH4_NEXT	**Size**	32 bits
Offset	0x144	**Read/Write**	Read/Write
Register address	0xB0000144	**Initial value**	Unknown

Bit Assignments of R_DMA_CH4_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.37 R_DMA_CH4_BUF

DMA Channel 4 Buffer Register, General Characteristics

ID of register	R_DMA_CH4_BUF	**Size**	32 bits
Offset	0x148	**Read/Write**	Read/Write
Register address	0xB0000148	**Initial value**	Unknown

Bit Assignments of R_DMA_CH4_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.38 R_DMA_CH4_FIRST

DMA Channel 4 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH4_FIRST	**Size**	32 bits
Offset	0x1B0	**Read/Write**	Read/Write
Register address	0xB00001B0	**Initial value**	Unknown

Bit Assignments of R_DMA_CH4_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.39 R_DMA_CH4_CMD

DMA Channel 4 Command Register, General Characteristics

ID of register	R_DMA_CH4_CMD	Size	8 bits
Offset	0x1E0	Read/Write	Read/Write
Register address	0xB00001E0	Initial value	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH4_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH4_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH4_DESCR. start: Start processing list at R_DMA_CH4_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH4_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.40 R_DMA_CH4_CLR_INTR

DMA Channel 4 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH4_CLR_INTR	**Size**	8 bits
Offset	0x1E1	**Read/Write**	Write only
Register address	0xB00001E1	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH4_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.41 R_DMA_CH4_STATUS

DMA Channel 4 Status Register, General Characteristics

ID of register	R_DMA_CH4_STATUS	**Size**	8 bits
Offset	0x1E2	**Read/Write**	Read only
Register address	0xB00001E2	**Initial value**	Unknown

Bit Assignments of R_DMA_CH4_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 4 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.42 R_DMA_CH5_HWSW

DMA Channel 5 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH5_HWSW	**Size**	32 bits
Offset	0x150	**Read/Write**	Read/Write
Register address	0xB0000150	**Initial value**	Unknown

Bit Assignments of R_DMA_CH5_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.43 R_DMA_CH5_DESCR

DMA Channel 5 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH5_DESCR	Size	32 bits
Offset	0x15C	Read/Write	Read/Write
Register address	0xB000015C	Initial value	Unknown

Bit Assignments of R_DMA_CH5_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.44 R_DMA_CH5_NEXT

DMA Channel 5 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH5_NEXT	**Size**	32 bits
Offset	0x154	**Read/Write**	Read/Write
Register address	0xB0000154	**Initial value**	Unknown

Bit Assignments of R_DMA_CH5_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.45 R_DMA_CH5_BUF

DMA Channel 5 Buffer Register, General Characteristics

ID of register	R_DMA_CH5_BUF	**Size**	32 bits
Offset	0x158	**Read/Write**	Read/Write
Register address	0xB0000158	**Initial value**	Unknown

Bit Assignments of R_DMA_CH5_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.46 R_DMA_CH5_FIRST

DMA Channel 5 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH5_FIRST	**Size**	32 bits
Offset	0x1B4	**Read/Write**	Read/Write
Register address	0xB00001B4	**Initial value**	Unknown

Bit Assignments of R_DMA_CH5_FIRST

Bit(s)	**Name**	**Description**	**State/Range**
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.47 R_DMA_CH5_CMD

DMA Channel 5 Command Register, General Characteristics

ID of register	R_DMA_CH5_CMD	**Size**	8 bits
Offset	0x1E4	**Read/Write**	Read/Write
Register address	0xB00001E4	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH5_CMD

Bit(s)	**Name**	**Description**	**State/Range**
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH5_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH5_DESCR. start: Start processing list at R_DMA_CH5_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH5_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.48 R_DMA_CH5_CLR_INTR

DMA Channel 5 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH5_CLR_INTR	**Size**	8 bits
Offset	0x1E5	**Read/Write**	Write only
Register address	0xB00001E5	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH5_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.49 R_DMA_CH5_STATUS

DMA Channel 5 Status Register, General Characteristics

ID of register	R_DMA_CH5_STATUS	**Size**	8 bits
Offset	0x1E6	**Read/Write**	Read only
Register address	0xB00001E6	**Initial value**	Unknown

Bit Assignments of R_DMA_CH5_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 5 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.50 R_DMA_CH6_HWSW

DMA Channel 6 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH6_HWSW	**Size**	32 bits
Offset	0x160	**Read/Write**	Read/Write
Register address	0xB0000160	**Initial value**	Unknown

Bit Assignments of R_DMA_CH6_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.51 R_DMA_CH6_DESCR

DMA Channel 6 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH6_DESCR	**Size**	32 bits
Offset	0x16C	**Read/Write**	Read/Write
Register address	0xB000016C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH6_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.52 R_DMA_CH6_NEXT

DMA Channel 6 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH6_NEXT	**Size**	32 bits
Offset	0x164	**Read/Write**	Read/Write
Register address	0xB0000164	**Initial value**	Unknown

Bit Assignments of R_DMA_CH6_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.53 R_DMA_CH6_BUF

DMA Channel 6 Buffer Register, General Characteristics

ID of register	R_DMA_CH6_BUF	**Size**	32 bits
Offset	0x168	**Read/Write**	Read/Write
Register address	0xB0000168	**Initial value**	Unknown

Bit Assignments of R_DMA_CH6_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.54 R_DMA_CH6_FIRST

DMA Channel 6 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH6_FIRST	Size	32 bits
Offset	0x1B8	Read/Write	Read/Write
Register address	0xB00001B8	Initial value	Unknown

Bit Assignments of R_DMA_CH6_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.55 R_DMA_CH6_CMD

DMA Channel 6 Command Register, General Characteristics

ID of register	R_DMA_CH6_CMD	**Size**	8 bits
Offset	0x1E8	**Read/Write**	Read/Write
Register address	0xB00001E8	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH6_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH6_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH6_DESCR. start: Start processing list at R_DMA_CH6_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH6_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.56 R_DMA_CH6_CLR_INTR

DMA Channel 6 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH6_CLR_INTR	**Size**	8 bits
Offset	0x1E9	**Read/Write**	Write only
Register address	0xB00001E9	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH6_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.57 R_DMA_CH6_STATUS

DMA Channel 6 Status Register, General Characteristics

ID of register	R_DMA_CH6_STATUS	**Size**	8 bits
Offset	0x1EA	**Read/Write**	Read only
Register address	0xB00001EA	**Initial value**	Unknown

Bit Assignments of R_DMA_CH6_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 6 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.58 R_DMA_CH7_HWSW

DMA Channel 7 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH7_HWSW	**Size**	32 bits
Offset	0x170	**Read/Write**	Read/Write
Register address	0xB0000170	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.59 R_DMA_CH7_DESCR

DMA Channel 7 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH7_DESCR	**Size**	32 bits
Offset	0x17C	**Read/Write**	Read/Write
Register address	0xB000017C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.60 R_DMA_CH7_NEXT

DMA Channel 7 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH7_NEXT	**Size**	32 bits
Offset	0x174	**Read/Write**	Read/Write
Register address	0xB0000174	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.61 R_DMA_CH7_BUF

DMA Channel 7 Buffer Register, General Characteristics

ID of register	R_DMA_CH7_BUF	**Size**	32 bits
Offset	0x178	**Read/Write**	Read/Write
Register address	0xB0000178	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.62 R_DMA_CH7_FIRST

DMA Channel 7 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH7_FIRST	**Size**	32 bits
Offset	0x1BC	**Read/Write**	Read/Write
Register address	0xB00001BC	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.63 R_DMA_CH7_CMD

DMA Channel 7 Command Register, General Characteristics

ID of register	R_DMA_CH7_CMD	**Size**	8 bits
Offset	0x1EC	**Read/Write**	Read/Write
Register address	0xB00001EC	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH7_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH7_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH7_DESCR. start: Start processing list at R_DMA_CH7_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH7_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.64 R_DMA_CH7_CLR_INTR

DMA Channel 7 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH7_CLR_INTR	**Size**	8 bits
Offset	0x1ED	**Read/Write**	Write only
Register address	0xB00001ED	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH7_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.65 R_DMA_CH7_STATUS

DMA Channel 7 Status Register, General Characteristics

ID of register	R_DMA_CH7_STATUS	**Size**	8 bits
Offset	0x1EE	**Read/Write**	Read only
Register address	0xB00001EE	**Initial value**	Unknown

Bit Assignments of R_DMA_CH7_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 7 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.66 R_DMA_CH8_HWSW

DMA Channel 8 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH8_HWSW	Size	32 bits
Offset	0x180	Read/Write	Read/Write
Register address	0xB0000180	Initial value	Unknown

Bit Assignments of R_DMA_CH8_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.67 R_DMA_CH8_DESCR

DMA Channel 8 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_DESCR	**Size**	32 bits
Offset	0x18C	**Read/Write**	Read/Write
Register address	0xB000018C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.68 R_DMA_CH8_NEXT

DMA Channel 8 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_NEXT	**Size**	32 bits
Offset	0x184	**Read/Write**	Read/Write
Register address	0xB0000184	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.69 R_DMA_CH8_BUF

DMA Channel 8 Buffer Register, General Characteristics

ID of register	R_DMA_CH8_BUF	**Size**	32 bits
Offset	0x188	**Read/Write**	Read/Write
Register address	0xB0000188	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.70 R_DMA_CH8_FIRST

DMA Channel 8 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_FIRST	**Size**	32 bits
Offset	0x1C0	**Read/Write**	Read/Write
Register address	0xB00001C0	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.71 R_DMA_CH8_CMD

DMA Channel 8 Command Register, General Characteristics

ID of register	R_DMA_CH8_CMD	**Size**	8 bits
Offset	0x1F0	**Read/Write**	Read/Write
Register address	0xB00001F0	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH8_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH8_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH8_DESCR. start: Start processing list at R_DMA_CH8_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH8_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.72 R_DMA_CH8_CLR_INTR

DMA Channel 8 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH8_CLR_INTR	**Size**	8 bits
Offset	0x1F1	**Read/Write**	Write only
Register address	0xB00001F1	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH8_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.73 R_DMA_CH8_STATUS

DMA Channel 8 Status Register, General Characteristics

ID of register	R_DMA_CH8_STATUS	**Size**	8 bits
Offset	0x1F2	**Read/Write**	Read only
Register address	0xB00001F2	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 8 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.14.74 R_DMA_CH8_SUB

DMA Channel 8 Subchannel Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_SUB	**Size**	32 bits
Offset	0x18C	**Read/Write**	Read/Write
Register address	0xB000018C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB

Bit(s)	Name	Description	State/Range
31 - 0	sub	This register gives the pointer to the current SB descriptor, and is shared between all subchannels.	

18.14.75 R_DMA_CH8_NEP

DMA Channel 8 Next Endpoint Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_NEP	**Size**	32 bits
Offset	0x1C0	**Read/Write**	Read/Write
Register address	0xB00001C0	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_NEP

Bit(s)	Name	Description	State/Range
31 - 0	nep	This register gives the pointer to the next endpoint descriptor. Endpoint descriptors must be 32-bit aligned for correct operation.	

18.14.76 R_DMA_CH8_SUB0_EP

DMA Channel 8 Subchannel 0 Endpoint Descriptor Register, General Characteristics

ID of register	R_DMA_CH8_SUB0_EP	**Size**	32 bits
Offset	0x1C8	**Read/Write**	Read/Write
Register address	0xB00001C8	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB0_EP

Bit(s)	Name	Description	State/Range
31 - 0	ep	This register gives the pointer to current endpoint descriptor for the subchannel. Endpoint descriptors must be 32-bit aligned for correct operation.	

18.14.77 R_DMA_CH8_SUB0_CMD

DMA Channel 8 Subchannel 0 Command Register, General Characteristics

ID of register	R_DMA_CH8_SUB0_CMD	**Size**	8 bits
Offset	0x1D3	**Read/Write**	Read/Write
Register address	0xB00001D3	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB0_CMD

Bit(s)	**Name**	**Description**	**State/Range**
7 - 1	Reserved	-	0
0	cmd	This is the subchannel command register. start: Starts the subchannel, but nothing will happen until the USB interface tells the subchannel what to do. There is no need to reset the FIFO before starting a subchannel since the DMA controller will reset the FIFO when changing to a new subchannel. stop: Stop the subchannel. However, the subchannel will finish current activities before stopping.	0 = stop 1 = start

18.14.78 R_DMA_CH8_SUB0_CLR_INTR

DMA Channel 8 Subchannel 0 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH8_SUB0_CLR_INTR	**Size**	8 bits
Offset	0x1E3	**Read/Write**	Write only
Register address	0xB00001E3	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB0_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	clr_descr	Setting this bit to do (1) clears the DMA channel 8, subchannel 0 descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.79 R_DMA_CH8_SUB1_EP

DMA Channel 8 Subchannel 1 Endpoint Register, General Characteristics

ID of register	R_DMA_CH8_SUB1_EP	**Size**	32 bits
Offset	0x1CC	**Read/Write**	Read/Write
Register address	0xB00001CC	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB1_EP

Bit(s)	Name	Description	State/Range
31 - 0	ep	This register gives the pointer to current endpoint descriptor for the subchannel. Endpoint descriptors must be 32-bit aligned for correct operation.	

18.14.80 R_DMA_CH8_SUB1_CMD

DMA Channel 8 Subchannel 1 Command Register, General Characteristics

ID of register	R_DMA_CH8_SUB1_CMD	**Size**	8 bits
Offset	0x1D7	**Read/Write**	Read/Write
Register address	0xB00001D7	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB1_CMD

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	cmd	This is the subchannel command register. start: Starts the subchannel, but nothing will happen until the USB tells the subchannel what to do. There is no need to reset the FIFO before starting a subchannel since the DMA controller will reset the FIFO when changing to a new subchannel. stop: Stop the subchannel. However, the subchannel will finish current activities before stopping.	0 = stop 1 = start

18.14.81 R_DMA_CH8_SUB1_CLR_INTR

DMA Channel 8 Subchannel 1 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH8_SUB1_CLR_INTR	**Size**	8 bits
Offset	0x1E7	**Read/Write**	Write only
Register address	0xB00001E7	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB1_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	clr_descr	Setting this bit to do (1) clears the dma channel 8, subchannel 1 descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.82 R_DMA_CH8_SUB2_EP

DMA Channel 8 Subchannel 2 Endpoint Register, General Characteristics

ID of register	R_DMA_CH8_SUB2_EP	**Size**	32 bits
Offset	0x1F8	**Read/Write**	Read/Write
Register address	0xB00001F8	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB2_EP

Bit(s)	Name	Description	State/Range
31 - 0	ep	This register gives the pointer to current endpoint descriptor for the subchannel. Endpoint descriptors must be 32-bit aligned for correct operation.	

18.14.83 R_DMA_CH8_SUB2_CMD

DMA Channel 8 Subchannel 2 Command Register, General Characteristics

ID of register	R_DMA_CH8_SUB2_CMD	Size	8 bits
Offset	0x1DB	Read/Write	Read/Write
Register address	0xB00001DB	Initial value	Unknown

Bit Assignments of R_DMA_CH8_SUB2_CMD

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	cmd	This is the subchannel command register. start: Starts the subchannel, but nothing will happen until the USB tells the subchannel what to do. There is no need to reset the FIFO before starting a subchannel since the DMA controller will reset the FIFO when changing to a new subchannel. stop: Stop the subchannel. However, the subchannel will finish current activities before stopping.	0 = stop 1 = start

18.14.84 R_DMA_CH8_SUB2_CLR_INTR

DMA Channel 8 Subchannel 2 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH8_SUB2_CLR_INTR	**Size**	8 bits
Offset	0x1EB	**Read/Write**	Write only
Register address	0xB00001EB	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB2_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	clr_descr	Setting this bit to do (1) clears the dma channel 8, subchannel 2 descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.85 R_DMA_CH8_SUB3_EP

DMA Channel 8 Subchannel 3 Endpoint Register, General Characteristics

ID of register	R_DMA_CH8_SUB3_EP	**Size**	32 bits
Offset	0x1FC	**Read/Write**	Read/Write
Register address	0xB00001FC	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB3_EP

Bit(s)	Name	Description	State/Range
31 - 0	ep	This register gives the pointer to current endpoint descriptor for the subchannel. Endpoint descriptors must be 32-bit aligned for correct operation.	

18.14.86 R_DMA_CH8_SUB3_CMD

DMA Channel 8 Subchannel 3 Command Register, General Characteristics

ID of register	R_DMA_CH8_SUB3_CMD	**Size**	8 bits
Offset	0x1DF	**Read/Write**	Read/Write
Register address	0xB00001DF	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB3_CMD

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	cmd	This is the subchannel command register. start: Starts the subchannel, but nothing will happen until the USB tells the subchannel what to do. There is no need to reset the FIFO before starting a subchannel since the DMA controller will reset the FIFO when changing to a new subchannel. stop: Stop the subchannel. However, the subchannel will finish current activities before stopping.	0 = stop 1 = start

18.14.87 R_DMA_CH8_SUB3_CLR_INTR

DMA Channel 8 Subchannel 3 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH8_SUB3_CLR_INTR	**Size**	8 bits
Offset	0x1EF	**Read/Write**	Write only
Register address	0xB00001EF	**Initial value**	Unknown

Bit Assignments of R_DMA_CH8_SUB3_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	clr_descr	Setting this bit to do (1) clears the dma channel 8, subchannel 3 descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.88 R_DMA_CH9_HWSW

DMA Channel 9 Hardware/Software Data Buffer Length Register, General Characteristics

ID of register	R_DMA_CH9_HWSW	Size	32 bits
Offset	0x190	Read/Write	Read/Write
Register address	0xB0000190	Initial value	Unknown

Bit Assignments of R_DMA_CH9_HWSW

Bit(s)	Name	Description	State/Range
31 - 16	hw	This field gives the current number of bytes left in the DMA buffer (note). hw is updated each time DMA accesses the DMA buffer.	0 - 65535
15 - 0	sw	This field gives the total length in bytes of the DMA buffer (note). sw is updated when a new descriptor is read by the DMA channel.	0 - 65535

Note: If all bits are 0, the length is 2^{16}.

18.14.89 R_DMA_CH9_DESCR

DMA Channel 9 Current Descriptor Register, General Characteristics

ID of register	R_DMA_CH9_DESCR	**Size**	32 bits
Offset	0x19C	**Read/Write**	Read/Write
Register address	0xB000019C	**Initial value**	Unknown

Bit Assignments of R_DMA_CH9_DESCR

Bit(s)	Name	Description	State/Range
31 - 0	descr	This field gives the pointer to the current descriptor for the DMA channel, and is updated just before a new descriptor is read. When DMA stops due to an end-of-list, descr is not updated allowing it to be restarted later.	

18.14.90 R_DMA_CH9_NEXT

DMA Channel 9 Next Descriptor Register, General Characteristics

ID of register	R_DMA_CH9_NEXT	**Size**	32 bits
Offset	0x194	**Read/Write**	Read/Write
Register address	0xB0000194	**Initial value**	Unknown

Bit Assignments of R_DMA_CH9_NEXT

Bit(s)	Name	Description	State/Range
31 - 0	next	This field gives the pointer to the next descriptor, and is updated when a new descriptor is read.	

18.14.91 R_DMA_CH9_BUF

DMA Channel 9 Buffer Register, General Characteristics

ID of register	R_DMA_CH9_BUF	**Size**	32 bits
Offset	0x198	**Read/Write**	Read/Write
Register address	0xB0000198	**Initial value**	Unknown

Bit Assignments of R_DMA_CH9_BUF

Bit(s)	Name	Description	State/Range
31 - 0	buf	This field gives the pointer to next position in the buffer DMA will access. It is updated as DMA accesses data in the buffer.	

18.14.92 R_DMA_CH9_FIRST

DMA Channel 9 First Descriptor Register, General Characteristics

ID of register	R_DMA_CH9_FIRST	**Size**	32 bits
Offset	0x1C4	**Read/Write**	Read/Write
Register address	0xB00001C4	**Initial value**	Unknown

Bit Assignments of R_DMA_CH9_FIRST

Bit(s)	Name	Description	State/Range
31 - 0	first	This field gives the pointer to the first descriptor in the packet currently processed by the DMA channel, and must be updated by software before starting the DMA channel. It is updated by DMA after it has accessed the packet for the last time, and before DMA advances to the next packet in the list. first is set to zero by the DMA channel when end-of-list is reached.	

18.14.93 R_DMA_CH9_CMD

DMA Channel 9 Command Register, General Characteristics

ID of register	R_DMA_CH9_CMD	**Size**	8 bits
Offset	0x1F4	**Read/Write**	Read/Write
Register address	0xB00001F4	**Initial value**	Bits 7 to 3 are unknown. Bits 2 to 0 are set to 0 at reset.

Bit Assignments of R_DMA_CH9_CMD

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2 - 0	cmd	This is the command register to control DMA operation. When a command is completed, the DMA channel clears this register (i.e. cmd is set to hold (0)) and stops. hold: Hold the DMA channel in its current state. The hold command is completed immediately. Note that the hold command will fail if the DMA channel has completed the previous command before the hold command is given. An unsuccessful hold command is indicated by: (1) If the DMA channel reached end-of-list, R_DMA_CH9_FIRST is zero. (2) If the DMA channel received stop-from-io, the stop bit is set in the descriptor at R_DMA_CH9_DESCR. start: Start processing list at R_DMA_CH9_FIRST. This command completes at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. restart: Restart after end-of-list has been reached. The DMA channel re-reads the descriptor at R_DMA_CH9_DESCR, and if the eol-bit is no longer set, it immediately follows the next link in the re- read descriptor ignoring the wait, intr, and eop bits. This command is completed at end-of-list (1), stop-from-io (2), or a reset command (3). If the DMA channel is already running this command is ignored. continue: Continue after a successful hold command. If the hold command was unsuccessful, the continue command will be interpreted as a restart command. The continue command completes when the command held by the hold command has completed. reset: Reset the DMA channel and its FIFOs. When the channel has won arbitration, the reset takes 100 ns to complete.	0 = hold 1 = start 3 = restart 3 = continue 4 = reset

18.14.94 R_DMA_CH9_CLR_INTR

DMA Channel 9 Clear Interrupt Register, General Characteristics

ID of register	R_DMA_CH9_CLR_INTR	**Size**	8 bits
Offset	0x1F5	**Read/Write**	Write only
Register address	0xB00001F5	**Initial value**	Not applicable

Bit Assignments of R_DMA_CH9_CLR_INTR

Bit(s)	Name	Description	State/Range
7 - 2	Reserved	-	0
1	clr_eop	Setting this bit to do (1) clears the eop interrupt.	0 = dont 1 = do
0	clr_descr	Setting this bit to do (1) clears the descriptor interrupt.	0 = dont 1 = do

Note: Interrupts corresponding to fields set to do (1) are cleared, their field values are not saved. Fields set to 0 are not affected.

18.14.95 R_DMA_CH9_STATUS

DMA Channel 9 Status Register, General Characteristics

ID of register	R_DMA_CH9_STATUS	**Size**	8 bits
Offset	0x1F6	**Read/Write**	Read only
Register address	0xB00001F6	**Initial value**	Unknown

Bit Assignments of R_DMA_CH9_STATUS

Bit(s)	Name	Description	State/Range
7	Reserved	-	0
6 - 0	avail	This field shows the number of bytes in the DMA channel 9 FIFO. If there is more than one packet in the FIFO, avail reflects the first packet put into the FIFO.	0 - 64

18.15 Test Mode Registers

18.15.1 R_TEST_MODE

Test Mode Register, General Characteristics

ID of register	R_TEST_MODE	**Size**	32 bits
Offset	0xFC	**Read/Write**	Write only
Register address	0xB00000FC	**Initial value**	1

Bit Assignments of R_TEST_MODE

Bit(s)	Name	Description	State/Range
31 - 20	Reserved	-	0
19	single_step	This field is used to enable and disable single step.	0 = off 1 = on
18	step_wr	This field is used to enable and disable single step break on memory write cycles.	0 = off 1 = on
17	step_rd	This field is used to enable and disable single step break on memory data read cycles.	0 = off 1 = on
16	step_fetch	This field is used to enable and disable single step break on instruction fetch cycles.	0 = off 1 = on
15 - 13	Reserved	-	0
12	mmu_test	This field is used to enable and disable the MMU test mode. In MMU test mode, the bus fault output to the CPU/cache is always zero.	0 = off 1 = on
11	usb_test	This field is used to enable and disable the USB test mode.	0 = off 1 = on
10	scsi_timer_test	This field is used to enable and disable the SCSI arbitration timer test mode.	0 = off 1 = on
9	backoff	This field is used to enable and disable the network transmitter backoff test mode. During loopback mode, this bit also enables the col signal.	0 = off 1 = on
8	snmp_test	This field is used to enable and disable the test mode for the Ethernet error and statistics counters.	0 = off 1 = on
7	snmp_inc	Ethernet error and statistics counter test mode increment clock. When snmp_test is set to on, a 0->1 transition makes all Ethernet error and statistics counters count up by one.	0 = dont 1 = do
6	ser_loop	Select loopback for the serial ports. Note that the loopback test only works if the baudrate bit in this register is set to off (0). This field controls the synchronous serial port loopback as well.	0 = off 1 = on
5	baudrate	This field is used to enable and disable the baudrate select test mode. When set, the selected baudrate clocks are output on the asynchronous and synchronous serial port $\overline{\text{rts}}$ and txd pins. The baudrate clocks are 20 ns long positive pulses.	0 = off 1 = on

Bit Assignments of R_TEST_MODE (continued)

Bit(s)	Name	Description	State/Range
4 - 3	timer	Timer clock divider test bits. The clock divider consists of a prescaler and four cascaded 4-bit counters. The test modes break up the carry between the 4-bit counters. off: Normal operation. even: Counters 0 and 2 have 50 MHZ input. Carry from counters 0 to 1 and from 2 to 3 is connected normally. odd: Counters 0, 1 and 3 have input from the prescaler (14.75 MHz). Carry from counters 2 to 3 is connected normally. all: All 4-bit counters have 50 MHz input. Please note that this is a test mode, and it is not intended for normal operation.	0 = off 1 = even 2 = odd 3 = all
2	cache_test	This field is used to enable and disable the cache memory test mode. If the cache memory test mode is enabled, and the cache is enabled in the cache_enable field below, all memory accesses are treated as cacheable and cache hits (i.e. 8 kbyte RAM cache).	0 = normal 1 = test

1	tag_test	This field is used to enable and disable the cache tag memory test mode. It is used primarily for initializing and production tests. In this mode, it is possible to write to the cache's tag memory. DMA accesses are not allowed in this mode.
		`if (addr[31] == 0){`
		`  TM.tag[addr[12:5]] = addr[30:13];`
		`  TM.dirty[addr[12:5]] = addr[0];`
		`}`
			0 = normal
			1 = test

0	cache_enable	This field is used to enable and disable the cache. If the cache is disabled, all memory accesses are treated as non-cacheable accesses and, as a result, bypass the cache.	0 = disable 1 = enable

18.15.2 R_SINGLE_STEP

Single Step Register, General Characteristics

ID of register	R_SINGLE_STEP	**Size**	8 bits
Offset	0xFE	**Read/Write**	Write only
Register address	0xB00000FE	**Initial value**	0

Bit Assignments of R_SINGLE_STEP

Bit(s)	Name	Description	State/Range
7 - 4	Reserved	-	0
3	single_step	This field is used to enable and disable single step.	0 = off 1 = on
2	step_wr	This field is used to enable and disable single step break on memory write cycles.	0 = off 1 = on
1	step_rd	This field is used to enable and disable single step break on memory data read cycles.	0 = off 1 = on
0	step_fetch	This field is used to enable and disable single step break on instruction fetch cycles.	0 = off 1 = on

Note: This 8-bit wide register is part of the R_TEST_MODE register.

18.16 Universal Serial Bus Interface Control Registers

Note: The initial values for all USB registers except R_USB_REVISION below are unknown until USB is enabled in R_GEN_CONFIG.

18.16.1 R_USB_REVISION

USB Revision Register, General Characteristics

ID of register	R_USB_REVISION	**Size**	8 bits
Offset	0x200	**Read/Write**	Read only
Register address	0xB0000200	**Initial value**	0x10 or 0x11 (note)

Bit Assignments of R_USB_REVISION

Bit(s)	Name	Description	State/Range
7 - 4	major	This field holds the major ETRAX 100LX USB internal revision number.	1
3 - 0	minor	This field holds the minor ETRAX 100LX USB internal revision number.	0 = v3 1 = v1, v2

Note: In revision 1 or 2 (v1,v2) of the ETRAX 100LX, this register shows a value of 0x11. In revision 3 (v3) of the ETRAX 100LX, this register reads 0x10. The USB version supported in both these cases is USB specification 1.1. The ETRAX 100LX v1 has part number 17511, v2 has part number 17854, and v3 has part number 18816.

18.16.2 R_USB_COMMAND

USB Command Register, General Characteristics

ID of register	R_USB_COMMAND	**Size**	8 bits
Offset	0x201	**Read/Write**	Read/Write
Register address	0xB0000201	**Initial value**	0x00

Bit Assignments of R_USB_COMMAND

Bit(s)	Name	Description	State/Range
7 - 6	port_sel	This field selects which port the port_cmd field is to be applied on.	0 = nop 1 = port1 2 = port2 3 = both
5 - 4	port_cmd	This field is the request to be sent to the port. Note that if either v2 or v3 of the ETRAX 100LX is used, port_cmd disable must be issued by using the register R_USB_PORT1_DISABLE for port 1 and R_USB_PORT2_DISABLE for port 2 instead of using R_USB_COMMAND.	0 = reset 1 = disable 2 = suspend 3 = resume
3	busy	This field is read only. It signals that the USB interface is busy executing the last given command. Any written value will be ignored.	0 = no 1 = yes
2 - 0	ctrl_cmd	This is the controller command.	0 = nop 1 = reset 2 = deconfig 3 = host_config 4 = dev_config 5 = host_reset 6 = host_run 7 = host_stop

Note: This register is used in host mode.

18.16.3 R_USB_COMMAND_DEV

USB Command Device Register, General Characteristics

ID of register	R_USB_COMMAND_DEV	**Size**	8 bits
Offset	0x201	**Read/Write**	Read/Write
Register address	0xB0000201	**Initial value**	0x00

Bit Assignments of R_USB_COMMAND_DEV

Bit(s)	Name	Description	State/Range
7 - 6	port_sel	This field selects which port the port_cmd field is applied on. It is not necessary to remember which port was configured in R_GEN_CONFIG. Use any (3) to command a port and nop (0) to not command a port.	0 = nop 3 = any
5 - 4	port_cmd	This field is the request to send to the port.	0 = active 1 = passive 2 = nop 3 = wakeup
3	busy	This field is read only. It signals that the USB interface is busy executing the last given command. Any written value will be ignored.	0 = no 1 = yes
2 - 0	ctrl_cmd	This is the controller command.	0 = nop 1 = reset 2 = deconfig 3 = host_config 4 = dev_config 5 = dev_active 6 = dev_passive 7 = dev_nop

Note: This register is used in device mode.

18.16.4 R_USB_STATUS

USB Status Register, General Characteristics

ID of register	R_USB_STATUS	**Size**	8 bits
Offset	0x202	**Read/Write**	Read only
Register address	0xB0000202	**Initial value**	0x00

Bit Assignments of R_USB_STATUS

Bit(s)	Name	Description	State/Range
7 - 6	Reserved	-	0
5	ourun	This field indicates that an overrun or underrun condition occurred. The bit is cleared by a read from R_USB_EPID_ATTN.	0 = no 1 = yes
4	perror	This field indicates that there was something wrong with the contents of a DMA descriptor, or that an invalid EPID was used. The bit is cleared by a read from R_USB_EPID_ATTN.	0 = no 1 = yes
3	device_mode	This field indicates that the USB interface operates in device mode.	0 = no 1 = yes
2	host_mode	This field indicates that the USB interface operates in host mode.	0 = no 1 = yes
1	started	In host mode, this field shows that the controller has been started. In device mode, it shows that the device is active.	0 = no 1 = yes
0	running	In host, mode this bit indicates that the host is running. In device mode, it is always no (0).	0 = no 1 = yes

18.16.5 R_USB_IRQ_MASK_SET

USB IRQ Mask Set Register, General Characteristics

ID of register	R_USB_IRQ_MASK_SET	**Size**	16 bits
Offset	0x204	**Read/Write**	Write only
Register address	0xB0000204	**Initial value**	Not applicable

Bit Assignments of R_USB_IRQ_MASK_SET

Bit(s)	Name	Description	State/Range
15 - 14	Reserved	-	0
13	iso_eof	This field sets the mask bit for the iso_eof interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
12	intr_eof	This field sets the mask bit for the intr_eof interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
11	iso_eot	This field sets the mask bit for the iso_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
10	intr_eot	This field sets the mask bit for the intr_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
9	ctl_eot	This field sets the mask bit for the ctl_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
8	bulk_eot	This field sets the mask bit for the bulk_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
7 - 4	Reserved	-	0
3	epid_attn	This field sets the mask bit for the epid_attn interrupt. The interrupt is cleared when register R_USB_EPID_ATTN is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
2	sof	This field sets the mask bit for the sof interrupt. The interrupt is cleared when register R_USB_FM_NUMBER is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
1	port_status	This field sets the mask bit for the port_status interrupt. The interrupt is cleared after changed port status registers R_USB_RH_PORT_STATUS_1 and R_USB_RH_PORT_STATUS_2 have been read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
0	ctl_status	This field sets the mask bit for the ctl_status interrupt. The interrupt is cleared when register R_USB_STATUS is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set

Note 1: The masks for interrupts corresponding to fields set to (1) are set, their field values are not saved. Fields set to nop (0) are not affected.

Note 2: This register is used in host mode.

18.16.6 R_USB_IRQ_MASK_READ

USB IRQ Mask Read Register, General Characteristics

ID of register	R_USB_IRQ_MASK_READ	Size	16 bits
Offset	0x204	Read/Write	Read only
Register address	0xB0000204	Initial value	0x00

Bit Assignments of R_USB_IRQ_MASK_READ

Bit(s)	Name	Description	State/Range
15-14	Reserved	-	0
13	iso_eof	This field shows the status of the iso_eof interrupt after the individual mask.	0 = no_pend 1 = pend
12	intr_eof	This field shows the status of the intr_eof interrupt after the individual mask.	0 = no_pend 1 = pend
11	iso_eot	This field shows the status of the iso_eot interrupt after the individual mask.	0 = no_pend 1 = pend
10	intr_eot	This field shows the status of the intr_eot interrupt after the individual mask.	0 = no_pend 1 = pend
9	ctl_eot	This field shows the status of the ctl_eot interrupt after the individual mask.	0 = no_pend 1 = pend
8	bulk_eot	This field shows the status of the bulk_eot interrupt after the individual mask.	0 = no_pend 1 = pend
7 - 4	Reserved	-	0
3	epid_attn	This field shows the status of the epid_attn interrupt after the individual mask.	0 = no_pend 1 = pend
2	sof	This field shows the status of the sof interrupt after the individual mask.	0 = no_pend 1 = pend
1	port_status	This field shows the status of the port_status interrupt after the individual mask.	0 = no_pend 1 = pend
0	ctl_status	This field shows the status of the ctl_status interrupt after the individual mask.	0 = no_pend 1 = pend

Note: This register is used in host mode.

18.16.7 R_USB_IRQ_MASK_CLR

USB IRQ Mask Clear Register, General Characteristics

ID of register	R_USB_IRQ_MASK_CLR	**Size**	16 bits
Offset	0x206	**Read/Write**	Write only
Register address	0xB0000206	**Initial value**	Not applicable

Bit Assignments of R_USB_IRQ_MASK_CLR

Bit(s)	Name	Description	State/Range
15 - 14	Reserved	-	0
13	iso_eof	This field clears the mask bit of the iso_eof interrupt.	0 = nop 1 = clr
12	intr_eof	This field clears the mask bit of the intr_eof interrupt.	0 = nop 1 = clr
11	iso_eot	This field clears the mask bit of the iso_eot interrupt.	0 = nop 1 = clr
10	intr_eot	This field clears the mask bit of the intr_eot interrupt.	0 = nop 1 = clr
9	ctl_eot	This field clears the mask bit of the ctl_eot interrupt.	0 = nop 1 = clr
8	bulk_eot	This field clears the mask bit of the bulk_eot interrupt.	0 = nop 1 = clr
7 - 4	Reserved	-	0
3	epid_attn	This field clears the mask bit of the epid_attn interrupt.	0 = nop 1 = clr
2	sof	This field clears the mask bit of the sof interrupt.	0 = nop 1 = clr
1	port_status	This field clears the mask bit of the port_status interrupt.	0 = nop 1 = clr
0	ctl_status	This field clears the mask bit of the ctl_status interrupt.	0 = nop 1 = clr

Note 1: The masks for interrupts corresponding to fields set to clr (1) are cleared, their field values are not saved. Fields set to nop (0) are not affected.

Note 2: This register is used in host mode.

18.16.8 R_USB_IRQ_READ

USB IRQ Read Register, General Characteristics

ID of register	R_USB_IRQ_READ	**Size**	16 bits
Offset	0x206	**Read/Write**	Read only
Register address	0xB0000206	**Initial value**	0x00

Bit Assignments of R_USB_IRQ_READ

Bit(s)	Name	Description	State/Range
15 - 14	Reserved	-	0
13	iso_eof	This field shows the status of the iso_eof interrupt prior to the individual mask.	0 = no_pend 1 = pend
12	intr_eof	This field shows the status of the intr_eof interrupt prior to the individual mask.	0 = no_pend 1 = pend
11	iso_eot	This field shows the status of the iso_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
10	intr_eot	This field shows the status of the intr_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
9	ctl_eot	This field shows the status of the ctl_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
8	bulk_eot	This field shows the status of the bulk_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
7 - 4	Reserved	-	0
3	epid_attn	This field shows the status of the epid_attn interrupt prior to the individual mask.	0 = no_pend 1 = pend
2	sof	This field shows the status of the sof interrupt prior to the individual mask.	0 = no_pend 1 = pend
1	port_status	This field shows the status of the port_status interrupt prior to the individual mask.	0 = no_pend 1 = pend
0	ctl_status	This field shows the status of the ctl_status interrupt prior to the individual mask.	0 = no_pend 1 = pend

Note: This register is used in host mode.

18.16.9 R_USB_IRQ_MASK_SET_DEV

USB IRQ Mask Set Device Register, General Characteristics

ID of register	R_USB_IRQ_MASK_SET_DEV	**Size**	16 bits
Offset	0x204	**Read/Write**	Write only
Register address	0xB0000204	**Initial value**	Not applicable

Bit Assignments of R_USB_IRQ_MASK_SET_DEV

Bit(s)	Name	Description	State/Range
15 - 13	Reserved	-	0
12	out_eot	This field sets the mask bit for the out_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
11	ep3_in_eot	This field sets the mask bit for the ep3_in_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
10	ep2_in_eot	This field sets the mask bit for the ep2_in_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
9	ep1_in_eot	This field sets the mask bit for the ep1_in_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
8	ep0_in_eot	This field sets the mask bit for the ep0_in_eot interrupt. The interrupt is cleared when the R_USB_EPID_ATTN register is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
7 - 4	Reserved	-	0
3	epid_attn	This field sets the mask bit for the epid_attn interrupt. The interrupt is cleared when register R_USB_EPID_ATTN is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
2	sof	This field sets the mask bit for the sof interrupt. The interrupt is cleared when R_USB_FM_NUMBER_DEV is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
1	port_status	This field sets the mask bit for the port_status interrupt. The interrupt is cleared after the R_USB_RH_PORT_STATUS_1 and R_USB_RH_PORT_STATUS_2 registers have been read. The interrupt has the vector number 0x3F.	0 = nop 1 = set
0	ctl_status	This field sets the mask bit for the ctl_status interrupt. The interrupt is cleared when R_USB_STATUS is read. The interrupt has the vector number 0x3F.	0 = nop 1 = set

Note 1: The masks for interrupts corresponding to fields set to (1) are set, their field values are not saved. Fields set to nop (0) are not affected.

Note 2: This register is used in device mode.

18.16.10 R_USB_IRQ_MASK_READ_DEV

USB IRQ Mask Read Device Register, General Characteristics

ID of register	R_USB_IRQ_MASK_READ_DEV	**Size**	16 bits
Offset	0x204	**Read/Write**	Read only
Register address	0xB0000204	**Initial value**	0x00

Bit Assignments of R_USB_IRQ_MASK_READ_DEV

Bit(s)	Name	Description	State/Range
15 - 13	Reserved	-	0
12	out_eot	This field shows the status of the out_eot interrupt after the individual mask.	0 = no_pend 1 = pend
11	ep3_in_eot	This field shows the status of the ep3_in_eot interrupt after the individual mask.	0 = no_pend 1 = pend
10	ep2_in_eot	This field shows the status of the ep2_in_eot interrupt after the individual mask.	0 = no_pend 1 = pend
9	ep1_in_eot	This field shows the status of the ep1_in_eot interrupt after the individual mask.	0 = no_pend 1 = pend
8	ep0_in_eot	This field shows the status of the ep0_in_eot interrupt after the individual mask.	0 = no_pend 1 = pend
7 - 4	Reserved	-	0
3	epid_attn	This field shows the status of the epid_attn interrupt after to the individual mask.	0 = no_pend 1 = pend
2	sof	This field shows the status of the sof interrupt after to the individual mask.	0 = no_pend 1 = pend
1	port_status	This field shows the status of the port_status interrupt after to the individual mask.	0 = no_pend 1 = pend
0	ctl_status	This field shows the status of the ctl_status interrupt after to the individual mask.	0 = no_pend 1 = pend

Note: This register is used in device mode.

18.16.11 R_USB_IRQ_MASK_CLR_DEV

USB IRQ Mask Clear Device Register, General Characteristics

ID of register	R_USB_IRQ_MASK_CLR_DEV	Size	16 bits
Offset	0x206	Read/Write	Write only
Register address	0xB0000206	Initial value	Not applicable

Bit Assignments of R_USB_IRQ_MASK_CLR_DEV

Bit(s)	Name	Description	State/Range
15 - 13	Reserved	-	0
12	out_eot	This field clears the interrupt mask bit for the out_eot interrupt.	0 = nop 1 = clr
11	ep3_in_eot	This field clears the interrupt mask bit for the ep3_in_eot interrupt.	0 = nop 1 = clr
10	ep2_in_eot	This field clears the interrupt mask bit for the ep2_in_eot interrupt.	0 = nop 1 = clr
9	ep1_in_eot	This field clears the interrupt mask bit for the ep1_in_eot interrupt.	0 = nop 1 = clr
8	ep0_in_eot	This field clears the interrupt mask bit for the ep0_in_eot interrupt.	0 = nop 1 = clr
7 - 4	Reserved	-	0
3	epid_attn	This field clears the interrupt mask bit for the epid_attn interrupt.	0 = nop 1 = clr
2	sof	This field clears the interrupt mask bit for the sof interrupt.	0 = nop 1 = clr
1	port_status	This field clears the interrupt mask bit for the port_status interrupt.	0 = nop 1 = clr
0	ctl_status	This field clears the interrupt mask bit for the ctl_status interrupt.	0 = nop 1 = clr

Note 1: The masks for interrupts corresponding to fields set to clr (1) are cleared, their field values are not saved. Fields set to nop (0) are not affected.

Note 2: This register is used in device mode.

18.16.12 R_USB_IRQ_READ_DEV

USB IRQ Read Device Register, General Characteristics

ID of register	R_USB_IRQ_READ_DEV	**Size**	16 bits
Offset	0x206	**Read/Write**	Read only
Register address	0xB0000206	**Initial value**	0x00

Bit Assignments of R_USB_IRQ_READ_DEV

Bit(s)	Name	Description	State/Range
15 - 13	Reserved	-	0
12	out_eot	This field shows the status for the out_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
11	ep3_in_eot	This field shows the status for the ep3_in_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
10	ep2_in_eot	This field shows the status for the ep2_in_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
9	ep1_in_eot	This field shows the status for the ep1_in_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
8	ep0_in_eot	This field shows the status for the ep0_in_eot interrupt prior to the individual mask.	0 = no_pend 1 = pend
7 - 4	Reserved	-	0
3	epid_attn	This field shows the status for the epid_attn interrupt prior to the individual mask.	0 = no_pend 1 = pend
2	sof	This field shows the status for the sof interrupt prior to the individual mask.	0 = no_pend 1 = pend
1	port_status	This field shows the status for the port_status interrupt prior to the individual mask.	0 = no_pend 1 = pend
0	ctl_status	This field shows the status for the ctl_status interrupt prior to the individual mask.	0 = no_pend 1 = pend

Note: This register is used in device mode.

18.16.13 R_USB_FM_NUMBER

USB Frame Number Register, General Characteristics

ID of register	R_USB_FM_NUMBER	**Size**	32 bits
Offset	0x20C	**Read/Write**	Read/Write
Register address	0xB000020C	**Initial value**	0x00000000

Bit Assignments of R_USB_FM_NUMBER

Bit(s)	Name	Description	State/Range
31 - 0	value	This register holds the value of the current USB frame number. It is cleared on USB controller reset.	

Note: This register is used in host mode.

18.16.14 R_USB_FM_NUMBER_DEV

USB Frame Number Device Register, General Characteristics

ID of register	R_USB_FM_NUMBER_DEV	**Size**	32 bits
Offset	0x20C	**Read/Write**	Read/Write
Register address	0xB000020C	**Initial value**	0x00000000

Bit Assignments of R_USB_FM_NUMBER_DEV

Bit(s)	Name	Description	State/Range
31	sign	If sign is equal to early (0), the host frame is deviation bits shorter than the standard USB frame size. If sign is equal to late (1), the host frame is deviation bits longer the standard USB frame size.	0 = early 1 = late
30 - 24	deviation		0 - 127
23 - 11	Reserved	-	0
10 - 0	fm_number	This field gives the latest received sof number.	0 - 1023

Note 1: This register is used in device mode to read the value of the current USB frame, and is cleared on USB controller reset.

Note 2: In device mode, software has to update the register R_USB_FM_INTERVAL if the sof IRQ must follow the SOF packet. By using the fields sign and deviation in R_USB_FM_NUMBER_DEV, software can decide how R_USB_FM_INTERVAL should be updated.

18.16.15 R_USB_FM_INTERVAL

USB Frame Interval Register, General Characteristics

ID of register	R_USB_FM_INTERVAL	**Size**	16 bits
Offset	0x210	**Read/Write**	Read/Write
Register address	0xB0000210	**Initial value**	0x2EDF

Bit Assignments of R_USB_FM_INTERVAL

Bit(s)	Name	Description	State/Range
15 - 14	Reserved	-	0
13 - 6	fixed	This field gives the upper eight bits of the frame interval. These bits are read only.	187
5 - 0	adj	This field gives the lower six bits of the frame interval. These bits can be written to in order to adjust the frame interval if needed. The standard says that the frame interval must not deviate from the nominal 12000 bit times by more than 15 bit times. Host controller software must not adjust the frame interval by more than one bit time over six frames, and only one bit time in each adjustment.	0 - 63

Note: This register contains a 14 bit value defining the bit time interval in a frame (i.e. the distance between two sofs). The frame timer counts from this value down to zero. The value is reloaded into the R_USB_FM_REMAINING register at each start of frame. The value in this register is the frame length minus one.

18.16.16 R_USB_FM_REMAINING

USB Frame Remaining Register, General Characteristics

ID of register	R_USB_FM_REMAINING	**Size**	16 bits
Offset	0x212	**Read/Write**	Read only
Register address	0xB0000212	**Initial value**	0x0000

Bit Assignments of R_USB_FM_REMAINING

Bit(s)	**Name**	**Description**	**State/Range**
15 - 14	Reserved	-	0
13 - 0	value	This register shows the remaining number of bit times in the current frame.	

18.16.17 R_USB_FM_PSTART

USB Frame Periodic Start Register, General Characteristics

ID of register	R_USB_FM_PSTART	**Size**	16 bits
Offset	0x214	**Read/Write**	Read/Write
Register address	0xB0000214	**Initial value**	0x0000

Bit Assignments of R_USB_FM_PSTART

Bit(s)	Name	Description	State/Range
15 - 14	Reserved	-	0
13 - 0	value	This register gives the periodic start point.	

Note: This register is used in host mode.

18.16.18 R_USB_RH_STATUS

USB Root Hub Status Register, General Characteristics

ID of register	R_USB_RH_STATUS	Size	8 bits
Offset	0x203	Read/Write	Read only
Register address	0xB0000203	Initial value	Undefined

Bit Assignments of R_USB_RH_STATUS

Bit(s)	Name	Description	State/Range
7	babble2	This bit is set if there is a babbling device detected on USB Port p2. It is reset by reading R_USB_RH_PORT_STATUS_2.	0 = no 1 = yes
6	babble1	This bit is set if there is a babbling device detected on USB Port p1. It is reset by reading R_USB_RH_PORT_STATUS_1.	0 = no 1 = yes
5 - 4	bus1	This field shows the bus state of port 1 as sampled at the EOF2 time in the frame, and is used for diagnostic purposes. If port 1 is not configured, then this field is undefined.	0 = SE0 1 = Diff0 1 = Diff1 3 = SE1
3 - 2	bus2	This field shows the bus state of port 2 as sampled at the EOF2 time in the frame, and is used for diagnostic purposes. If port 2 is not configured, then this field is undefined.	0 = SE0 1 = Diff0 1 = Diff1 3 = SE1
1 - 0	nports	This field shows the number of root port hubs. It is controlled by the USB bits in R_GEN_CONFIG. It is only possible to read two values: 1 and 2. If no ports are configured, the USB interface will be turned off, rending the values undefined.	1 - 2

18.16.19 R_USB_RH_PORT_STATUS_1

USB Root Hub Port Status 1 Register, General Characteristics

ID of register	R_USB_RH_PORT_STATUS_1	**Size**	16 bits
Offset	0x218	**Read/Write**	Read only
Register address	0xB0000218	**Initial value**	0x0000

Bit Assignments of R_USB_RH_PORT_STATUS_1

Bit(s)	Name	Description	State/Range
15 - 10	Reserved	-	0
9	speed	This field shows the speed of the connected device. full (0) = Full speed device low (1) = Low speed device	0 = full 1 = low
8	power	This field is not implemented in hardware, and is always read as zero.	0 = no
7 - 5	Reserved	-	0
4	reset	This field is set during the reset sequence of this port	0 = no 1 = yes
3	overcurrent	This field is not implemented in hardware, and is always read as zero.	0 = no
2	suspended	This field tells whether or not the connected device is suspended.	0 = no
1	enabled	This field shows the status of whether the port is enabled or disabled for USB traffic.	0 = no 1 = yes
0	connected	This field shows if there is a device connected to this port.	0 = no 1 = yes

Note: This register is compatible with the wPortStatus field of the get_status command to the USB hubs (These terms relate to the Universal Serial Bus specification Rev 1.1). Two bits of wPortStatus are not implemented in hardware and must be implemented in software. These are the overcurrent and power fields. The reading of R_USB_RH_PORT_STATUS_1, together with R_USB_RH_PORT_STATUS_2, clears the root hub status change interrupt condition.

18.16.20 R_USB_RH_PORT_STATUS_2

USB Root Hub Port Status 2 Register, General Characteristics

ID of register	R_USB_RH_PORT_STATUS_2	**Size**	16 bits
Offset	0x21A	**Read/Write**	Read only
Register address	0xB000021A	**Initial value**	0x0000

Bit Assignments of R_USB_RH_PORT_STATUS_2

Bit(s)	Name	Description	State/Range
15 - 10	Reserved	-	0
9	speed	This field shows the speed of the connected device. full (0) = Full speed device low (1) = Low speed device	0 = full 1 = low
8	power	This field is not implemented in hardware, and is always read as zero.	0 = no
7 - 5	Reserved	-	0
4	reset	This field is set during the reset sequence of this port	0 = no 1 = yes
3	overcurrent	This field is not implemented in hardware, and is always read as zero.	0 = no
2	suspended	This field tells whether or not the connected device is suspended.	0 = no 1 = yes
1	enabled	This field shows the status of whether the port is enabled or disabled for USB traffic.	0 = no 1 = yes
0	connected	This field shows if there is a device connected to this port.	0 = no 1 = yes

Note: This register is compatible with the wPortStatus field of the get_status command to the USB hubs (These terms relate to the Universal Serial Bus specification Rev 1.1). Two bits of wPortStatus are not implemented in hardware and must be implemented in software. These are the overcurrent and power fields. The reading of R_USB_RH_PORT_STATUS_2, together with R_USB_RH_PORT_STATUS_1, clears the root hub status change interrupt condition.

18.16.21 R_USB_EPT_INDEX

USB End Point Table Index Register, General Characteristics

ID of register	R_USB_EPT_INDEX	**Size**	8 bits
Offset	0x208	**Read/Write**	Read/Write
Register address	0xB0000208	**Initial value**	0x00

Bit Assignments of R_USB_EPT_INDEX

Bit(s)	Name	Description	State/Range
7 - 5	Reserved	-	0
4 - 0	value	The index into the endpoint lookup table to be used when reading and writing to the R_USB_EPT_DATA register. The endpoint lookup table contains 32 endpoint entries, each pointing at an endpoint on a device on the USB. The table is indexed by the endpoint ID (EPID) number in the USB DMA descriptors.	0 - 31

18.16.22 R_USB_EPT_DATA

USB End Point Table Data Register, General Characteristics

ID of register	R_USB_EPT_DATA	Size	32 bits
Offset	0x21C	Read/Write	Read/Write
Register address	0xB000021C	Initial value	0x00000000

Bit Assignments of R_USB_EPT_DATA

Bit(s)	Name	Description	State/Range
31	valid	This field indicates the validity of this table entry.	0 = no 1 = yes
30	hold	This field is used to tell software to stay off of this entry. Hardware will use this entry for more transactions. This field may be cleared by software when traffic for this endpoint is stopped.	0 = no 1 = yes
29 - 28	error_count_in	This field is the transaction error counter for IN transactions. Even though this field is writable, it should only be set to zero by software.	0 - 3
27	t_in	This is the data toggle bit for IN transactions.	0 - 1
26	low_speed	This field marks the endpoint as a low speed endpoint. This puts some protocol restrictions on what software can do with the endpoint. See the USB specification rev 1.1 for more information on low speed devices.	0 = no 1 = yes
25 - 24	port	This field indicates the expected port where upstream traffic for this device is due to occur. Downstream traffic goes to both ports.	0 = any 1 = p1 2 = p2 3 = undef
23 - 22	error_code	This field indicates what kind of error caused the endpoint to be disabled. A stall is not really a protocol error. It merely says that the endpoint gave a stall response. A bus_error is when two devices respond to a transaction request. This condition has to be resolved by software. buffer_error indicates a DMA overrun or underrun.	0 = no_error 1 = stall 2 = bus_error 3 = buffer_error
21	t_out	This is the toggle bit for OUT and SETUP transactions.	0 - 1
20 - 19	error_count_out	This is the transaction error counter for OUT and SETUP transactions. Even though this field is writable, it should only be set to zero by software.	0 - 3
18	Reserved	-	0
17 - 11	max_len	This is the max packet length for normal (i.e. not isochronous) data packets.	1 - 64
10 - 7	ep	This field gives the endpoint number.	0 - 15
6 - 0	dev	This field gives the configured device address.	0 - 127

Note: This is the general endpoint table data register. It is used for host mode normal transfers. For isochronous transfers in host mode see R_USB_EPT_DATA_ISO. For device mode, see R_USB_EPT_DATA_DEV. All these registers have the same address.

18.16.23 R_USB_EPT_DATA_ISO

USB End Point Table Data Isochronous Register, General Characteristics

ID of register	R_USB_EPT_DATA_ISO	**Size**	32 bits
Offset	0x21C	**Read/Write**	Read/Write
Register address	0xB000021C	**Initial value**	0x00000000

Bit Assignments of R_USB_EPT_DATA_ISO

Bit(s)	Name	Description	State/Range
31	valid	This field indicates the validity of this table entry.	0 = no 1 = yes
30 - 26	Reserved	-	0
25 - 24	port	This field indicates the expected port where upstream traffic for this device is due to occur. Downstream traffic goes to both ports.	0 = any 1 = p1 2 = p2 3 = undef
23 - 22	error_code	This field indicates what kind of error caused the endpoint to be disabled. (note 2)	0 = no_error 2 = bus_error 3 = buffer_error
21	Reserved	-	0
20 - 11	max_len	This field is the max packet length for isochronous data packets.	1 - 1023
10 - 7	ep	This field gives the endpoint number.	
6 - 0	dev	This field gives the configured device address.	

Note 1: This register is used for host mode isochronous endpoints.

18.16.24 R_USB_EPT_DATA_DEV

USB End Point Table Data Device Register, General Characteristics

ID of register	R_USB_EPT_DATA_DEV	**Size**	32 bits
Offset	0x21C	**Read/Write**	Read/Write
Register address	0xB000021C	**Initial value**	0x00000000

Bit Assignments of R_USB_EPT_DATA_DEV

Bit(s)	Name	Description	State/Range
31	valid	This field indicates the validity of this table entry.	0 = no 1 = yes
30	hold	This field is used to tell software to stay off of this entry. Hardware will use it for more transactions.	0 = no 1 = yes
29	stall	This field stalls this endpoint. The device will respond with a stall-packet to all requests from the host for this endpoint.	0 = no 1 = yes
28	iso_resp	This field is only used for isochronous endpoints. If this field is quiet (0), do not send a broken packet in case DMA is unable to deliver data. If iso_resp is yes (1), start to send packet; if data is unavailable, send an abort.	0 = quiet 1 = yes
27	ctrl	This field indicates that this is a control endpoint.	0 = no 1 = yes
26	iso	This field tells whether or not the endpoint is isochronous.	0 = no 1 = yes
25 - 24	Reserved	-	0
23	Reserved	-	0
22	Reserved	-	0
21	t	This is the toggle bit for device mode.	0 - 1
20 - 11	max_len	This field is the max packet length for this endpoint. Note that there are constraints in the protocol on the max packet length.	1 - 1023
10 - 7	ep	This field gives the endpoint number.	0 - 15
6 - 0	dev	This field gives the configured device address.	0 - 127

Note: This register is used in device mode.

18.16.25 R_USB_EPID_ATTN

USB End Point ID Attention Register, General Characteristics

ID of register	R_USB_EPID_ATTN	**Size**	32 bits
Offset	0x224	**Read/Write**	Read only
Register address	0xB0000224	**Initial value**	0

Bit Assignments of R_USB_EPID_ATTN

Bit(s)	Name	Description	State/Range
31 - 0	value	This register indicates which entries in the endpoint table have new information for software. Reading this register clears the ourun, perror, iso_eof, intr_eof, iso_eot, intr_eot, ctl_eot, bulk_eot, epid_attn, sof, port_status, ctl_status, out_eot, ep3_in_eot, ep3_in_eot, ep3_in_eot, ep3_in_eot, epid_attn interrupt conditions. This field shows the indicator bits, one per EPID.	

18.16.26 R_USB_PORT1_DISABLE

USB Port1 Disable Register, General Characteristics

ID of register	R_USB_PORT1_DISABLE	**Size**	8 bits
Offset	0x6A	**Read/Write**	Write only
Register address	0xB000006A	**Initial value**	Unknown

Bit Assignments of R_USB_PORT1_DISABLE

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	disable	This register is used to force USB Port p1 to do a disconnect in order to emulate a port disable. To disable a port, software sets R_USB_PORT1_DISABLE to yes (0), and waits for R_USB_RH_PORT_STATUS_1 to show enabled = no (0).	0 = yes 1 = no

Note: This register is only defined for the ETRAX 100LX v2 and v3.

18.16.27 R_USB_PORT2_DISABLE

USB Port2 Disable Register, General Characteristics

ID of register	R_USB_PORT2_DISABLE	**Size**	8 bits
Offset	0x52	**Read/Write**	Write only
Register address	0xB0000052	**Initial value**	Unknown

Bit Assignments of R_USB_PORT2_DISABLE

Bit(s)	Name	Description	State/Range
7 - 1	Reserved	-	0
0	disable	This register is used to force USB Port p2 to do a disconnect in order to emulate a port disable. To disable a port, software sets R_USB_PORT2_DISABLE to yes (0), and waits for R_USB_RH_PORT_STATUS_2 to show enabled = no (0).	0 = yes 1 = no

Note: This register is only defined for the ETRAX 100LX v2 and v3.

18.17 MMU Registers

18.17.1 R_MMU_CONFIG

MMU Configuration Register, General Characteristics.

ID of register	R_MMU_CONFIG	**Size**	32 bits
Offset	0x240	**Read/Write**	Write only
Address	0xB0000240	**Initial value**	0

Bit Assignments of R_MMU_CONFIG

Bit(s)	Name	Description	State/Range
31	mmu_enable	This field enables or disables the MMU.	0 = disable 1 = enable
30 - 19	Reserved	-	0
18	inv_excp	This field enables and disables the MMU invalid page exception. When inv_excp is enabled, a matching TLB entry with the valid bit cleared generates an invalid page exception. When inv_excp is disabled, a TLB entry with the valid bit cleared is treated as a miss exception. inv_excp should be enabled only when the valid bit is used for functions such as reference counting.	0 = disable 1 = enable
17	acc_excp	This bit enables and disables the protection of kernel pages during the user mode. When acc_excp is disabled, the kernel bit in the TLB entry is ignored and cannot generate an access violation exception. All kernel pages can then be referenced in user mode as well.	0 = disable 1 = enable
16	we_excp	This bit enables and disables the write protection of pages for which the we bit in the TLB entry is disabled. When we_excp is disabled, the we bit in the TLB entry is ignored and cannot generate a write error exception. All pages are thus write enabled.	0 = disable 1 = enable
15	seg_f	In kernel mode, this field selects linear segment or page mapping for segment f.	0 = page 1 = seg
14	seg_e	In kernel mode, this field selects linear segment or page mapping for segment e.	0 = page 1 = seg
13	seg_d	In kernel mode, this field selects linear segment or page mapping for segment d.	0 = page 1 = seg
12	seg_c	In kernel mode, this field selects linear segment or page mapping for segment c.	0 = page 1 = seg
11	seg_b	In kernel mode, this field selects linear segment or page mapping for segment b.	0 = page 1 = seg
10	seg_a	In kernel mode, this field selects linear segment or page mapping for segment a.	0 = page 1 = seg
9	seg_9	In kernel mode, this field selects linear segment or page mapping for segment 9.	0 = page 1 = seg
8	seg_8	In kernel mode, this field selects linear segment or page mapping for segment 8.	0 = page 1 = seg

Bit Assignments of R_MMU_CONFIG (continued)

Bit(s)	Name	Description	State/Range
7	seg_7	In kernel mode, this field selects linear segment or page mapping for segment 7.	0 = page 1 = seg
6	seg_6	In kernel mode, this field selects linear segment or page mapping for segment 6.	0 = page 1 = seg
5	seg_5	In kernel mode, this field selects linear segment or page mapping for segment 5.	0 = page 1 = seg
4	seg_4	In kernel mode, this field selects linear segment or page mapping for segment 4.	0 = page 1 = seg
3	seg_3	In kernel mode, this field selects linear segment or page mapping for segment 3.	0 = page 1 = seg
2	seg_2	In kernel mode, this field selects linear segment or page mapping for segment 2.	0 = page 1 = seg
1	seg_1	In kernel mode, this field selects linear segment or page mapping for segment 1.	0 = page 1 = seg
0	seg_0	In kernel mode, this field selects linear segment or page mapping for segment 0.	0 = page 1 = seg

18.17.2 R_MMU_KSEG

MMU Kernel Segment Register, General Characteristics

ID of register	R_MMU_KSEG	**Size**	16 bits
Offset	0x240	**Read/Write**	Write only
Address	0xB0000240	**Initial value**	0

Bit Assignments of R_MMU_KSEG

Bit(s)	Name	Description	State/Range
15	seg_f	In kernel mode, this field selects linear segment or page mapping for segment f.	0 = page 1 = seg
14	seg_e	In kernel mode, this field selects linear segment or page mapping for segment e.	0 = page 1 = seg
13	seg_d	In kernel mode, this field selects linear segment or page mapping for segment d.	0 = page 1 = seg
12	seg_c	In kernel mode, this field selects linear segment or page mapping for segment c.	0 = page 1 = seg
11	seg_b	In kernel mode, this field selects linear segment or page mapping for segment b.	0 = page 1 = seg
10	seg_a	In kernel mode, this field selects linear segment or page mapping for segment a.	0 = page 1 = seg
9	seg_9	In kernel mode, this field selects linear segment or page mapping for segment 9.	0 = page 1 = seg
8	seg_8	In kernel mode, this field selects linear segment or page mapping for segment 8.	0 = page 1 = seg
7	seg_7	In kernel mode, this field selects linear segment or page mapping for segment 7.	0 = page 1 = seg
6	seg_6	In kernel mode, this field selects linear segment or page mapping for segment 6.	0 = page 1 = seg
5	seg_5	In kernel mode, this field selects linear segment or page mapping for segment 5.	0 = page 1 = seg
4	seg_4	In kernel mode, this field selects linear segment or page mapping for segment 4.	0 = page 1 = seg
3	seg_3	In kernel mode, this field selects linear segment or page mapping for segment 3.	0 = page 1 = seg
2	seg_2	In kernel mode, this field selects linear segment or page mapping for segment 2.	0 = page 1 = seg
1	seg_1	In kernel mode, this field selects linear segment or page mapping for segment 1.	0 = page 1 = seg
0	seg_0	In kernel mode, this field selects linear segment or page mapping for segment 0.	0 = page 1 = seg

Note: R_MMU_KSEG is a 16-bit register that is part of configuration register R_MMU_CONFIG.

18.17.3 R_MMU_CTRL

MMU Control Register, General Characteristics

ID of register	R_MMU_CTRL	**Size**	8 bits
Offset	0x242	**Read/Write**	Write only
Address	0xB0000242	**Initial value**	0

Bit Assignments of R_MMU_CTRL

Bit(s)	Name	Description	State/Range
7 - 3	Reserved	-	0
2	inv_excp	This field enables and disables the MMU invalid page exception. When inv_excp is enabled, a matching TLB entry with the valid bit cleared generates an invalid page exception. When inv_excp is disabled, a TLB entry with the valid bit cleared is treated as a miss exception. inv_excp should be enabled only when the valid bit is used for functions such as reference counting.	0 = disable 1 = enable
1	acc_excp	This field enables and disables the protection of kernel pages during the user mode. When acc_excp is disabled, the kernel bit in the TLB entry is ignored and cannot generate an access violation exception. All kernel pages can then be referenced in user mode as well.	0 = disable 1 = enable
0	we_excp	This field enables and disables the write protection of pages for which the we bit in the TLB entry is disabled. When we_excp is disabled, the we bit in the TLB entry is ignored and cannot generate a write error exception. All pages are thus write enabled.	0 = disable 1 = enable

Note: R_MMU_CTRL is a byte (8-bit) register that is part of configuration register R_MMU_CONFIG.

18.17.4 R_MMU_ENABLE

MMU Enable Register, General Characteristics

ID of register	R_MMU_ENABLE	**Size**	8 bits
Offset	0x243	**Read/Write**	Write only
Address	0xB0000243	**Initial value**	0

Bit Assignments of R_MMU_ENABLE

Bit(s)	Name	Description	State/Range
7	mmu_enable	This bit enables or disables the MMU.	0 = disable 1 = enable
6 - 0	Reserved	-	0

Note: R_MMU_ENABLE is a byte (8-bit) register that is part of configuration register R_MMU_CONFIG.

18.17.5 R_MMU_KBASE_LO

MMU Kernel Base Low Register, General Characteristics

ID of register	R_MMU_KBASE_LO	**Size**	32 bits
Offset	0x244	**Read/Write**	Write only
Address	0xB0000244	**Initial value**	Unknown

Bit Assignments of R_MMU_KBASE_LO

Bit(s)	Name	Description	State/Range
31 - 28	base_7	The value of these field selects the kernel base for segment 7.	0 - 15
27 - 24	base_6	The value of these field selects the kernel base for segment 6.	0 - 15
23 - 20	base_5	The value of these field selects the kernel base for segment 5.	0 - 15
19 - 16	base_4	The value of these field selects the kernel base for segment 4.	0 - 15
15 - 12	base_3	The value of these field selects the kernel base for segment 3.	0 - 15
11 - 8	base_2	The value of these field selects the kernel base for segment 2.	0 - 15
7 - 4	base_1	The value of these field selects the kernel base for segment 1.	0 - 15
3 - 0	base_0	The value of these field selects the kernel base for segment 0.	0 - 15

18.17.6 R_MMU_KBASE_HI

MMU Kernel Base High Register, General Characteristics

ID of register	R_MMU_KBASE_HI	**Size**	32 bits
Offset	0x248	**Read/Write**	Write only
Address	0xB0000248	**Initial value**	Unknown

Bit Assignments of R_MMU_KBASE_HI

Bit(s)	Name	Description	State/Range
31 - 28	base_f	The value of these field selects the kernel base for segment f.	0 - 15
27 - 24	base_e	The value of these field selects the kernel base for segment e.	0 - 15
23 - 20	base_d	The value of these field selects the kernel base for segment d.	0 - 15
19 - 16	base_c	The value of these field selects the kernel base for segment c.	0 - 15
15 - 12	base_b	The value of these field selects the kernel base for segment b.	0 - 15
11 - 8	base_a	The value of these field selects the kernel base for segment a.	0 - 15
7 - 4	base_9	The value of these field selects the kernel base for segment 9.	0 - 15
3 - 0	base_8	The value of these field selects the kernel base for segment 8.	0 - 15

18.17.7 R_MMU_CONTEXT

MMU Context Register, General Characteristics

ID of register	R_MMU_CONTEXT	**Size**	8 bits
Offset	0x24C	**Read/Write**	Read/Write
Address	0xB000024C	**Initial value**	Unknown

Bit Assignments of R_MMU_CONTEXT

Bit(s)	**Name**	**Description**	**State/Range**
7 - 6	Reserved	-	0
5 - 0	page_id	The value of this field is the page identification of the current context. A page can only be referenced if the page_id field in this register matches the page_id field in the TLB entry, unless the global bit in the TLB entry is set.	0 - 63

18.17.8 R_MMU_CAUSE

MMU Cause Register, General Characteristics

ID of register	R_MMU_CAUSE	**Size**	32 bits
Offset	0x250	**Read/Write**	Read only
Address	0xB0000250	**Initial value**	Unknown

Bit Assignments of R_MMU_CAUSE

Bit(s)	Name	Description	State/Range
31 - 13	vpn	This field represents a virtual page number. When an MMU exception occurs, this field is updated with the vpn of the referenced address that generated the exception.	
12	miss_excp	This field indicates the occurrence of a miss exception. When set, the field signifies that the referenced address did not match any TLB entry. A valid entry must be loaded by software.	0 = no 1 = yes
11	inv_excp	This field indicates the occurrence of an invalid page exception. When set, the field signifies that reference was made to a page with matching vpn and page_id fields in the TLB, but the valid bit in the TLB entry was cleared. The inv_excp bit can be used for reference counting.	0 = no 1 = yes
10	acc_excp	This field indicates the occurrence of an access violation exception. When set, the field signifies that a reference from user mode was made to a page with the kernel bit set in the TLB entry. This is used to protect mapped kernel pages from user mode references.	0 = no 1 = yes
9	we_excp	This field indicates the occurrence of a write error exception. When set, the field signifies that during a write operation, reference was made to a page for which the we bit in the TLB entry was cleared. This exception can be used for both write protection and dirty checks.	0 = no 1 = yes
8	wr_rd	This field is updated when a memory management exception occurs. It indicates whether the exception was caused by a write or read access.	0 = read 1 = write
7 - 6	Reserved	-	0
5 - 0	page_id	This field represents a page identification. The field is updated when an MMU exception occurs, and holds the content of the page_id in R_MMU_CONTEXT.	0 - 63

Note: This register is also used to store the contents of R_TLB_HI, and its contents are destroyed when writing to R_TLB_HI. Register R_MMU_CAUSE is updated when an MMU exception occurs, identifying the cause of the exception. When R_TLB_LO is written, the page_id and vpn fields of R_MMU_CAUSE are written into the TLB using the index field in register R_TLB_SELECT.

18.17.9 R_TLB_SELECT

MMU TLB Select Register, General Characteristics

ID of register	R_TLB_SELECT	**Size**	8 bits
Offset	254	**Read/Write**	Read/Write
Address	0xB0000254	**Initial value**	Unknown

Bit Assignments of R_TLB_SELECT

Bit(s)	Name	Description	State/Range
7 - 6	Reserved	-	0
5 - 0	index	The value of this field represents the TLB index. The index field selects the TLB entry to use when registers R_TLB_LO and R_TLB_HI are used. In the event of a miss exception, the index is loaded with a random value which selects the entry to replace. The random value is always a valid index for the faulting virtual address. All other exceptions load the index field with a pointer to the entry that triggered the exception.	0 - 63

18.17.10 R_TLB_LO

MMU TLB Low Register, General Characteristics

ID of register	R_TLB_LO	**Size**	32 bits
Offset	0x258	**Read/Write**	Read/Write
Address	0xB0000258	**Initial value**	Unknown

Bit Assignments of R_TLB_LO

Bit(s)	Name	Description	State/Range
31 - 13	pfn	This field represents a physical frame number pfn.	
12 - 4	Reserved	-	0
3	global	This field is the global bit of the TLB entry.	0 = no 1 = yes
2	valid	This field is the valid bit of the TLB entry.	0 = no 1 = yes
1	kernel	This field is the kernel bit of the TLB entry.	0 = no 1 = yes
0	we	This field is the we bit of the TLB entry.	0 = no 1 = yes

Note: R_TLB_LO is used for reading and writing the lower part of an entry in the TLB. When this register is read, the pfn, global, valid, kernel and we fields of the TLB entry selected by the index field in R_TLB_SELECT will be read. When writing to R_TLB_LO, this value selected by the index field plus the page_id and vpn fields in R_MMU_CAUSE, are written into the TLB entry selected by the index field in R_TLB_SELECT.

18.17.11 R_TLB_HI

MMU TLB High Register, General Characteristics

ID of register	R_TLB_HI	**Size**	32 bits
Offset	0x25C	**Read/Write**	Read/Write
Address	0xB000025C	**Initial value**	Unknown

Bit Assignments of R_TLB_HI

Bit(s)	Name	Description	State/Range
31 - 13	vpn	This field gives the virtual page number.	
12 - 6	Reserved	-	0
5 - 0	page_id	This field gives the page identification number.	0 - 63

Note: R_TLB_HI is used for reading and writing the high part of an entry in the TLB. When this register is read, the page_id and vpn fields of the TLB entry selected by the index field in R_TLB_SELECT will be read. When writing to R_TLB_HI, this value selected by the index field is stored in the corresponding fields of R_MMU_CAUSE. When R_TLB_LO is written, the fields in R_MMU_CAUSE will be written into the TLB. The previous contents of R_MMU_CAUSE will be destroyed when writing to R_TLB_HI.

18.18 Synchronous Serial Port Registers

18.18.1 R_SYNC_SERIAL1_REC_DATA

Synchronous Serial Port 1 Receive Data Register, General Characteristics

ID of register	R_SYNC_SERIAL1_REC_DATA	**Size**	32 bits
Offset	0x6C	**Read/Write**	Read only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_REC_DATA

Bit(s)	Name	Description	State/Range
31 - 0	data_in	This field contains 32 bits of data from the Synchronous Serial Port p1 receiver.	

18.18.2 R_SYNC_SERIAL1_REC_WORD

Synchronous Serial Port 1 Receive 16-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL1_REC_WORD	**Size**	16 bits
Offset	0x6C	**Read/Write**	Read only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_REC_WORD

Bit(s)	Name	Description	State/Range
15 - 0	data_in	This field contains 16 bits of data in from the Synchronous Serial Port p1 receiver. The state/range for data_in depends on the wordsize field in “R_SYNC_SERIAL1_CTRL”.	0 - 65535

Note: This 16-bit wide register is part of the 32-bit R_SYNC_SERIAL1_REC_DATA register.

18.18.3 R_SYNC_SERIAL1_REC_BYTE

Synchronous Serial Port 1 Receive Byte Register, General Characteristics

ID of register	R_SYNC_SERIAL1_REC_BYTE	**Size**	8 bits
Offset	0x6C	**Read/Write**	Read only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_REC_BYTE

Bit(s)	Name	Description	State/Range
7 - 0	data_in	This field contains a byte of data in from the Synchronous Serial Port p1 receiver.	0 - 255

Note: This 8-bit wide register is part of the 32-bit R_SYNC_SERIAL1_REC_DATA register.

18.18.4 R_SYNC_SERIAL1_STATUS

Synchronous Serial Port 1 Status Register, General Characteristics

ID of register	R_SYNC_SERIAL1_STATUS	**Size**	32 bits
Offset	0x68	**Read/Write**	Read only
Register address	0xB0000068	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_STATUS

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	0
15	rec_status	This field indicates whether the Synchronous Serial Port p1 receiver is running. If so, data may be arriving which may generate interrupts or activate the DMA channel.	0 = running 1 = idle
14	tr_empty	This field indicates whether the transmitter data pipeline of Synchronous Serial Port p1 is empty. When set to empty (1), nothing more is sent unless new data is written to the output register or transferred by DMA to the transmitter.	1 = empty 0 = not_empty
13	tr_ready	This field indicates whether the transmitter of Synchronous Serial Port p1 is ready. When set to ready (1), either 32, 16 or 8 bits of data can be written to the serial transmitter.	0 = full 1 = ready
12	pin_1	This field is the value on input from ss1_in2. (note)	0 = low 1 = high
11	pin_0	This field is the value on input from ss1_in1. (note)	0 = low 1 = high
10	underflow	This field is set when an underflow error is detected in the transmitter of Synchronous Serial Port p1. The field is cleared when the register R_SYNC_SERIAL1_STATUS is read.	0 = no 1 = yes
9	overrun	This field is set when an overrun error is detected in the Synchronous Serial Port p1 receiver. The field is cleared when the register R_SYNC_SERIAL1_STATUS is read.	0 = no 1 = yes
8	data_avail	This field is set when data is available from the Synchronous Serial Port p1 receiver. The bit is cleared when the R_SYNC_SERIAL1_REC_DATA register is read.	0 = no 1 = yes
7 - 0	Reserved	-	Unknown

Note: For more information about pin usage see chapter *12 Synchronous Serial Interface*.

18.18.5 R_SYNC_SERIAL1_TR_DATA

Synchronous Serial Port 1 Transmit 32-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL1_TR_DATA	**Size**	32 bits
Offset	0x6C	**Read/Write**	Write only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_TR_DATA

Bit(s)	Name	Description	State/Range
31 - 0	data_out	This field gives 32 bits of data to the transmitter of Synchronous Serial Port p1.	

18.18.6 R_SYNC_SERIAL1_TR_WORD

Synchronous Serial Port 1 Transmit 16-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL1_TR_WORD	**Size**	16 bits
Offset	0x6C	**Read/Write**	Write only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_TR_WORD

Bit(s)	Name	Description	State/Range
15 - 0	data_out	This field contains 16 bits of data in from the Synchronous Serial Port p1 receiver. The state/range for data_out depends on the wordsize field in "R_SYNC_SERIAL1_CTRL".	0 - 65535

Note: This 16-bit wide register is part of the 32-bit R_SYNC_SERIAL1_TR_DATA register.

18.18.7 R_SYNC_SERIAL1_TR_BYTE

Synchronous Serial Port 1 Transmit Byte Register, General Characteristics

ID of register	R_SYNC_SERIAL1_TR_BYTE	**Size**	8 bits
Offset	0x6C	**Read/Write**	Write only
Register address	0xB000006C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL1_TR_BYTE

Bit(s)	Name	Description	State/Range
7 - 0	data_out	This field gives a byte of data to the transmitter of Synchronous Serial Port p1.	0 - 255

Note: This 8-bit wide register is part of the 32-bit R_SYNC_SERIAL1_TR_DATA register.

18.18.8 R_SYNC_SERIAL1_CTRL

Synchronous Serial Port 1 Control Register, General Characteristics

ID of register	R_SYNC_SERIAL1_CTRL	**Size**	32 bits
Offset	0x68	**Read/Write**	Write only
Register address	0xB0000068	**Initial value**	Bit 14 and 22 set to 0. Other bits unknown

Bit Assignments of R_SYNC_SERIAL1_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	The value in this field is the bit clock if baudrate clock is selected in register R_SYNC_SERIAL_PRESCALE.	0 = c150Hz 1 = c300Hz 2 = c600Hz 3 = c1200Hz 4 = c2400Hz 5 = c4800Hz 6 = c9600Hz 7 = c19k2Hz 8 = c28k8Hz 9 = c57k6Hz 10 = c115k2Hz 11 = c230k4Hz 12 = c460k8Hz 13 = c921k6Hz 14 = c3125kHz 15 = reserved
27	dma_enable	This field determines whether Synchronous Serial Port p1 should transfer data with DMA or by CPU accesses.	0 = off 1 = on
26 - 24	mode	The value in this field represents the mode selection command to Synchronous Serial Port p1: master - ETRAX generates clock and frame signals. slave - ETRAX listens to clock and frame signals. input - ETRAX receives data. output - ETRAX transmits data. bidir - ETRAX receives and transmits data. Note that Synchronous Serial Port p1 pin usage is changed when the value of the mode field is changed. Refer to chapter *13 Synchronous Serial Interface* more information.	0 = master_output 1 = slave_output 2 = master_input 3 = slave_input 4 = master_bidir 5 = slave_bidir
23	error	This field determines whether transfer and receiver errors are ignored for Synchronous Serial Port p1: When set to normal (0), transmission is stopped when an underflow or overrun condition is detected. When set to ignore (1), transfer and receiver errors are ignored. The underflow and overrun fields are set, but the transmission is not halted.	0 = normal 1 = ignore
22	rec_enable	This field enables or disables incoming data.	0 = disable 1 = enable
21	f_synctype	This field determines frame sync activity for Synchronous Serial Port p1: normal - the frame sync signal is active during the first bit of the word. early - the frame sync signal is active 1 bit before the first bit of the word.	0 = normal 1 = early

Bit Assignments of R_SYNC_SERIAL1_CTRL (continued)

Bit(s)	Name	Description	State/Range
20 - 19	f_syncsize	This field determines frame sync size for Synchronous Serial Port p1: bit - the frame sync signal is active during first bit of the word. word - the frame sync signal is active over the entire word. extended - the frame sync signal is active over the entire word plus 1 bit. When sending a non-interrupted stream of data, frame sync will be continuously high.	0 = bit 1 = word 2 = extended 3 = reserved
18	f_sync	This field enables or disables frame sync for Synchronous Serial Port p1: on - The frame sync is enabled. off - The frame sync is ignored. Incoming and outgoing data are treated as a bitstream.	0 = on 1 = off
17	clk_mode	This field selects the clock mode for Synchronous Serial Port p1: normal - The clock is running continuously. gated - The clock is turned off when no data is transmitted.	0 = normal 1 = gated
16	clk_halt	This field stops the clock for Synchronous Serial Port p1. When set to stopped (1), the frame sync generator is stopped.	0 = running 1 = stopped
15	bitorder	This field chooses whether the lsb or the msb is sent first for Synchronous Serial Port p1.	0 = lsb 1 = msb
14	tr_enable	This field enables or disables outgoing data for Synchronous Serial Port p1.	0 = disable 1 = enable
13 - 11	wordsize	This field selects data unit size for Synchronous Serial Port p1.	0 = size8bit 1 = size12bit 2 = size16bit 3 = size24bit 4 = size32bit
10	buf_empty	This field is the buffer empty flow control status indicator for Synchronous Serial Port p1. ss1status is active if no more than 0 or 8 bytes remain in the output DMA FIFO buffer.	0 = lmt_8 1 = lmt_0
9	buf_full	This field is the buffer full flow control status indicator for Synchronous Serial Port p1. ss1status is active if no more than 32 or 8 bytes of storage are free in the input DMA FIFO buffer.	0 = lmt_32 1 = lmt_8
8	flow_ctrl	This field enables or disables flow control for Synchronous Serial Port p1. If the handling of status input and output signals becomes congested, frame or word start can be delayed when this field is set to enabled (1).	0 = disabled 1 = enabled
7	Reserved	-	0
6	clk_polarity	This field selects the polarity of the sample edge of the incoming clock for Synchronous Serial Port p1.	0 = pos 1 = neg
5	frame_polarity	This field selects the polarity of the incoming frame signal for Synchronous Serial Port p1. When set to normal (0), the frame signal is active high.	0 = normal 1 = inverted
4	status_polarity	This field selects the polarity of the incoming status signal for Synchronous Serial Port p1. When set to normal (0), the status signal is active high.	0 = normal 1 = inverted

Bit Assignments of R_SYNC_SERIAL1_CTRL (continued)

Bit(s)	Name	Description	State/Range
3	clk_driver	This field selects the polarity of the outgoing clock signal for Synchronous Serial Port p1: normal - the internal reference clock signal is connected directly to the clock output. inverted - the internal reference clock signal is inverted and connected to the clock output.	0 = normal 1 = inverted
2	frame_driver	This field selects the polarity of the outgoing frame signal for Synchronous Serial Port p1: normal - the internal reference frame signal is connected directly to the status output. inverted - the internal reference frame signal is inverted and connected to the frame output.	0 = normal 1 = inverted
1	status_driver	This field selects the polarity of the outgoing status signal for Synchronous Serial Port p1: normal - the internal reference status signal is connected directly to the status output. inverted - the internal reference status signal is inverted and connected to the status output.	0 = normal 1 = inverted
0	def_out0	This field is the value of the ss1_out1 output pin when Serial Port p1 is enabled, but a sync serial mode is selected where the pin has no meaning/function (it is a spare pin given the configuration).	0 = low 1 = high

18.18.9 R_SYNC_SERIAL3_REC_DATA

Synchronous Serial Port 3 Receive 32-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL3_REC_DATA	**Size**	32 bits
Offset	0x7C	**Read/Write**	Read only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_REC_DATA

Bit(s)	Name	Description	State/Range
31 - 0	data_in	This field contains 32 bits of data from the Synchronous Serial Port p3 receiver.	

18.18.10 R_SYNC_SERIAL3_REC_WORD

Synchronous Serial Port 3 Receive 16-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL3_REC_WORD	**Size**	16 bits
Offset	0x7C	**Read/Write**	Read only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_REC_WORD

Bit(s)	Name	Description	State/Range
15 - 0	data_in	This field contains 16 bits of data from the Synchronous Serial Port p3 receiver. The state/range for data_in depends on the wordsize field in “R_SYNC_SERIAL3_CTRL”.	0 - 65535

Note: This 16-bit wide register is part of the 32-bit R_SYNC_SERIAL3_REC_DATA register.

18.18.11 R_SYNC_SERIAL3_REC_BYTE

Synchronous Serial Port 3 Receive Byte Register, General Characteristics

ID of register	R_SYNC_SERIAL3_REC_BYTE	**Size**	8 bits
Offset	0x7C	**Read/Write**	Read only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_REC_BYTE

Bit(s)	Name	Description	State/Range
7 - 0	data_in	This field contains a byte of data in from the Synchronous Serial Port p3 receiver.	0 - 255

Note: This 8-bit wide register is part of the 32-bit R_SYNC_SERIAL3_REC_DATA register.

18.18.12 R_SYNC_SERIAL3_STATUS

Synchronous Serial Port 3 Status Register, General Characteristics

ID of register	R_SYNC_SERIAL3_STATUS	**Size**	32 bits
Offset	0x78	**Read/Write**	Read only
Register address	0xB0000078	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_STATUS

Bit(s)	Name	Description	State/Range
31 - 16	Reserved	-	0
15	rec_status	This field indicates whether the Synchronous Serial Port p3 receiver is running. If so, data may be arriving which may generate interrupts or activate the DMA channel.	0 = running 1 = idle
14	tr_empty	This field indicates whether the transmitter data pipeline of Synchronous Serial Port p3 is empty. When set to empty (1), nothing more is sent unless new data is written to the output register or transferred by DMA to the transmitter.	1 = empty 0 = not_empty
13	tr_ready	This field indicates whether the transmitter of Synchronous Serial Port p3 is ready. When set to ready (1), either 32, 16 or 8 bits of data can be written to the serial transmitter.	0 = full 1 = ready
12	pin_1	This field is the value on input from ss3_in2. (note)	0 = low 1 = high
11	pin_0	This field is the value on input from ss3_in1. (note)	0 = low 1 = high
10	underflow	This field is set when an underflow error is detected in the transmitter of Synchronous Serial Port p3. The field is cleared when the register R_SYNC_SERIAL3_STATUS is read.	0 = no 1 = yes
9	overrun	This field is set when an overrun error is detected in the Synchronous Serial Port p3 receiver. The field is cleared when the register R_SYNC_SERIAL3_STATUS is read.	0 = no 1 = yes
8	data_avail	This field is set when data is available from the Synchronous Serial Port p3 receiver. The bit is cleared when the register R_SYNC_SERIAL3_REC_DATA is read.	0 = no 1 = yes
7 - 0	Reserved	-	0

Note: For more information about pin usage see chapter *13 Synchronous Serial Interface*.

18.18.13 R_SYNC_SERIAL3_TR_DATA

Synchronous Serial Port 3 Transmit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL3_TR_DATA	**Size**	32 bits
Offset	0x7C	**Read/Write**	Write only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_TR_DATA

Bit(s)	Name	Description	State/Range
31 - 0	data_out	This field gives 32 bits of data to the transmitter of Synchronous Serial Port p3.	

18.18.14 R_SYNC_SERIAL3_TR_WORD

Synchronous Serial Port 3 Transmit 16-bit Data Register, General Characteristics

ID of register	R_SYNC_SERIAL3_TR_WORD	**Size**	16 bits
Offset	0x7C	**Read/Write**	Write only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_TR_WORD

Bit(s)	Name	Description	State/Range
15 - 0	data_out	This field gives 16 bits of data to the transmitter of Synchronous Serial Port p3.	0 - 65535

Note: This 16-bit wide register is part of the 32-bit R_SYNC_SERIAL3_TR_DATA register.

18.18.15 R_SYNC_SERIAL3_TR_BYTE

Synchronous Serial Port 3 Transmit Byte Register, General Characteristics

ID of register	R_SYNC_SERIAL3_TR_BYTE	**Size**	8 bits
Offset	0x7C	**Read/Write**	Write only
Register address	0xB000007C	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL3_TR_BYTE

Bit(s)	Name	Description	State/Range
7 - 0	data_out	This field gives a byte of data to the transmitter of Synchronous Serial Port p3.	0 - 255

Note: This 8-bit wide register is part of the 32-bit R_SYNC_SERIAL3_TR_DATA register.

18.18.16 R_SYNC_SERIAL3_CTRL

Synchronous Serial Port 3 Control Register, General Characteristics

ID of register	R_SYNC_SERIAL3_CTRL	**Size**	32 bits
Offset	0x78	**Read/Write**	Write only
Register address	0xB0000078	**Initial value**	Bits 14 and 22 are set to 0. Other bits unknown

Bit Assignments of R_SYNC_SERIAL3_CTRL

Bit(s)	Name	Description	State/Range
31 - 28	tr_baud	The value in this field is the bit clock if baudrate clock is selected in register R_SYNC_SERIAL_PRESCALE.	0 = c150Hz 1 = c300Hz 2 = c600Hz 3 = c1200Hz 4 = c2400Hz 5 = c4800Hz 6 = c9600Hz 7 = c19k2Hz 8 = c28k8Hz 9 = c57k6Hz 10 = c115k2Hz 11 = c230k4Hz 12 = c460k8Hz 13 = c921k6Hz 14 = c3125kHz 15 = reserved
27	dma_enable	This field determines whether Synchronous Serial Port p3 should transfer data with DMA or by CPU accesses.	0 = off 1 = on
26 - 24	mode	The value in this field represents the mode selection command to Synchronous Serial Port p3: master - ETRAX generates clock and frame signals. slave - ETRAX listens to clock and frame signals. input - ETRAX receives data. output - ETRAX transmits data. bidir - ETRAX receives and transmits data. Note that Synchronous Serial Port p3 pin usage is changed when the value of the mode field is changed. Refer to chapter *13 Synchronous Serial Interface* for more information.	0 = master_output 1 = slave_output 2 = master_input 3 = slave_input 4 = master_bidir 5 = slave_bidir
23	error	This field determines whether transfer and receiver errors are ignored for Synchronous Serial Port p3: When set to normal (0), transmission is stopped when an underflow or overrun condition is detected. When set to ignore (1), transfer and receiver errors are ignored. The underflow and overrun fields are set, but the transmission is not halted.	0 = normal 1 = ignore
22	rec_enable	This field enables or disables incoming data.	0 = disable 1 = enable
21	f_synctype	This field determines frame sync activity for Synchronous Serial Port p3: normal - the frame sync signal is active during the first bit of the word. early - the frame sync signal is active 1 bit before the first bit of the word.	0 = normal 1 = early

Bit Assignments of R_SYNC_SERIAL3_CTRL (continued)

Bit(s)	Name	Description	State/Range
20 - 19	f_syncsize	This field determines frame sync size for Synchronous Serial Port p3: bit - the frame sync signal is active during first bit of the word. word - the frame sync signal is active over the entire word. extended - the frame sync signal is active over the entire word plus 1 bit. When sending a non-interrupted stream of data, frame sync will be continuously high.	0 = bit 1 = word 2 = extended 3 = reserved
18	f_sync	This field enables or disables frame sync for Synchronous Serial Port p3: on - The frame sync is enabled. off - The frame sync is ignored. Incoming and outgoing data are treated as a bitstream.	0 = on 1 = off
17	clk_mode	This field selects the clock mode for Synchronous Serial Port p3: normal - The clock is running continuously. gated - The clock is turned off when no data is transmitted.	0 = normal 1 = gated
16	clk_halt	This field stops the clock for Synchronous Serial Port p3. When set to stopped (1), the frame sync generator is temporarily stopped.	0 = running 1 = stopped
15	bitorder	This field chooses whether the lsb or the msb is sent first for Synchronous Serial Port p3.	0 = lsb 1 = msb
14	tr_enable	This field enables or disables outgoing data for Synchronous Serial Port p3.	0 = disable 1 = enable
13 - 11	wordsize	This field selects data unit size for Synchronous Serial Port p3.	0 = size8bit 1 = size12bit 2 = size16bit 3 = size24bit 4 = size32bit
10	buf_empty	This field is the buffer empty flow control status indicator for Synchronous Serial Port p3. ss3status is active if no more than 0 or 8 bytes remain in the output DMA FIFO buffer.	0 = lmt_8 1 = lmt_0
9	buf_full	This field is the buffer full flow control status indicator for Synchronous Serial Port p3. ss3status is active if no more than 32 or 8 bytes of storage are free in the input DMA FIFO buffer.	0 = lmt_32 1 = lmt_8
8	flow_ctrl	This field enables or disables flow control for Synchronous Serial Port p3. If the handling of status input and output signals becomes congested, frame or word start can be delayed when this field is set to enabled (1).	0 = disabled 1 = enabled
7	Reserved	-	0
6	clk_polarity	This field selects the polarity of the sample edge of the incoming clock for Synchronous Serial Port p3. If the negative edge is to be used, this field should be set to neg (1).	0 = pos 1 = neg
5	frame_polarity	This field selects the polarity of the incoming frame signal for Synchronous Serial Port p3. When set to normal (0), the frame signal is active high.	0 = normal 1 = inverted
4	status_polarity	This field selects the polarity of the incoming status signal for Synchronous Serial Port p3. When set to normal (0), the status signal is active high.	0 = normal 1 = inverted

Bit Assignments of R_SYNC_SERIAL3_CTRL (continued)

Bit(s)	Name	Description	State/Range
3	clk_driver	This field selects the polarity of the outgoing clock signal for Synchronous Serial Port p3: normal - the internal reference clock signal is connected directly to the clock output. inverted - the internal reference clock signal is inverted and connected to the clock output.	0 = normal 1 = inverted
2	frame_driver	This field selects the polarity of the outgoing frame signal for Synchronous Serial Port p3: normal - the internal reference frame signal is connected directly to the status output. inverted - the internal reference frame signal is inverted and connected to the frame output.	0 = normal 1 = inverted
1	status_driver	This field selects the polarity of the outgoing status signal for Synchronous Serial Port p3: normal - the internal reference status signal is connected directly to the status output. inverted - the internal reference status signal is inverted and connected to the status output.	0 = normal 1 = inverted
0	def_out0	This field is the value of the ss3_out1 output pin when Serial Port p3 is enabled, but a sync serial mode is selected where the pin has no meaning/function (it is a spare pin given the configuration).	0 = low 1 = high

18.18.17 R_SYNC_SERIAL_PRESCALE

Synchronous Serial Prescale Register, General Characteristics

ID of register	R_SYNC_SERIAL_PRESCALE	**Size**	32 bits
Offset	0xF4	**Read/Write**	Write only
Address	0xB00000F4	**Initial value**	Unknown

Bit Assignments of R_SYNC_SERIAL_PRESCALE

Bit(s)	Name	Description	State/Range
31 - 24	Reserved	-	0
23	clk_sel_u3	This field selects whether the codec clock or the baudrate clock is used as clock source for Synchronous Serial Port p3. If set to baudrate (1), the clock is generated from the baudrate clock from the asynchronous serial port. If set, the codec clock will be used. The source for the codec clock can be external or internal, defined by what operation mode is selected in R_SYNC_SERIAL3_CTRL.	0 = codec 1 = baudrate
22	word_stb_sel_u3	This field selects how the incoming word strobe for Synchronous Serial Port p3 is generated. If set, the word strobe is equal to the frame strobe and is extracted from the external incoming frame. This is the normal setting. If this field is set to internal (1), the word strobe is generated by internal counters. word_stb_sel_u3 should only be set to internal if the interface is running in any of the slave modes that are selected in R_SYNC_SERIAL3_CTRL.	0 = external 1 = internal
21	clk_sel_u1	This field selects whether the codec clock or the baudrate clock is used as clock source for Synchronous Serial Port p1. If set to baudrate (1), the clock is generated from the baudrate clock from the asynchronous serial port. If clk_sel_u3 is set to codec (0), the codec clock will be used. The source for the codec clock can be external or internal, defined by what operation mode is selected in R_SYNC_SERIAL1_CTRL.	0 = codec 1 = baudrate
20	word_stb_sel_u1	This field selects how the incoming word strobe Synchronous Serial Port p1 is generated. If set, the word strobe is equal to the frame strobe and is extracted from the external incoming frame. This is the normal setting. If this field is set to internal (1), the word strobe is generated by internal counters. word_stb_sel_u1 should only be set to internal if the interface is running in any of the slave modes that are selected in R_SYNC_SERIAL1_CTRL.	0 = external 1 = internal
19	Reserved	-	0
18 - 16	prescaler	This field sets the divide factor for the codec clock. Both synchronous serial ports are affected. The codec clock is 4.096 MHz divided by the division factor of 1 to 128. (note)	0 = div1 1 = div2 2 = div4 3 = div8 4 = div16 5 = div32 6 = div64 7 = div128
15	warp_mode	If warp_mode is enabled the codec base clock is changed from 4.096 MHz to 12.5 MHz. This is only used for testing purposes.	0 = normal 1 = enabled

Bit Assignments of R_SYNC_SERIAL_PRESCALE (continued)

14 - 11	frame_rate	This field selects the frame_trigger divisor. (note)	0-15
10	Reserved	-	0
9 - 0	word_rate	This field selects the word rate. Enough time must be left for the selected number of data bits to be transferred. If 8-bit word size is selected, the smallest value for this field is 7. (8 if early framesync is selected). (note)	0-1023

Note: In master mode, the bitclock and framesync rates are programable. The sample rate is selected by writing to this register which contains all integer division factors.

The bitclock is generated by dividing the internal ETRAX 100LX codec clock (4.096MHz) by (2^N). The valid range for the divisor is: 1 <= (2^N) <= 128. (See the prescaler field.)

Word sync is generated by dividing the bitclock by (D + 1) where D is the value of the word_rate field. The divisor may be in the range: 0 <= D <= 1023.

Framesync is generated by dividing wordsync by (F + 1) where F is the value of the frame_rate field. The divisor may be in the range: 0 <= D <= 15.

If the bitclock = 2.048 MHz (N=1) and word_rate = 255 and frame_rate = 0, then framesync and wordsync will be 8.0 KHz.

19 Electrical Information

19.1 Pinout

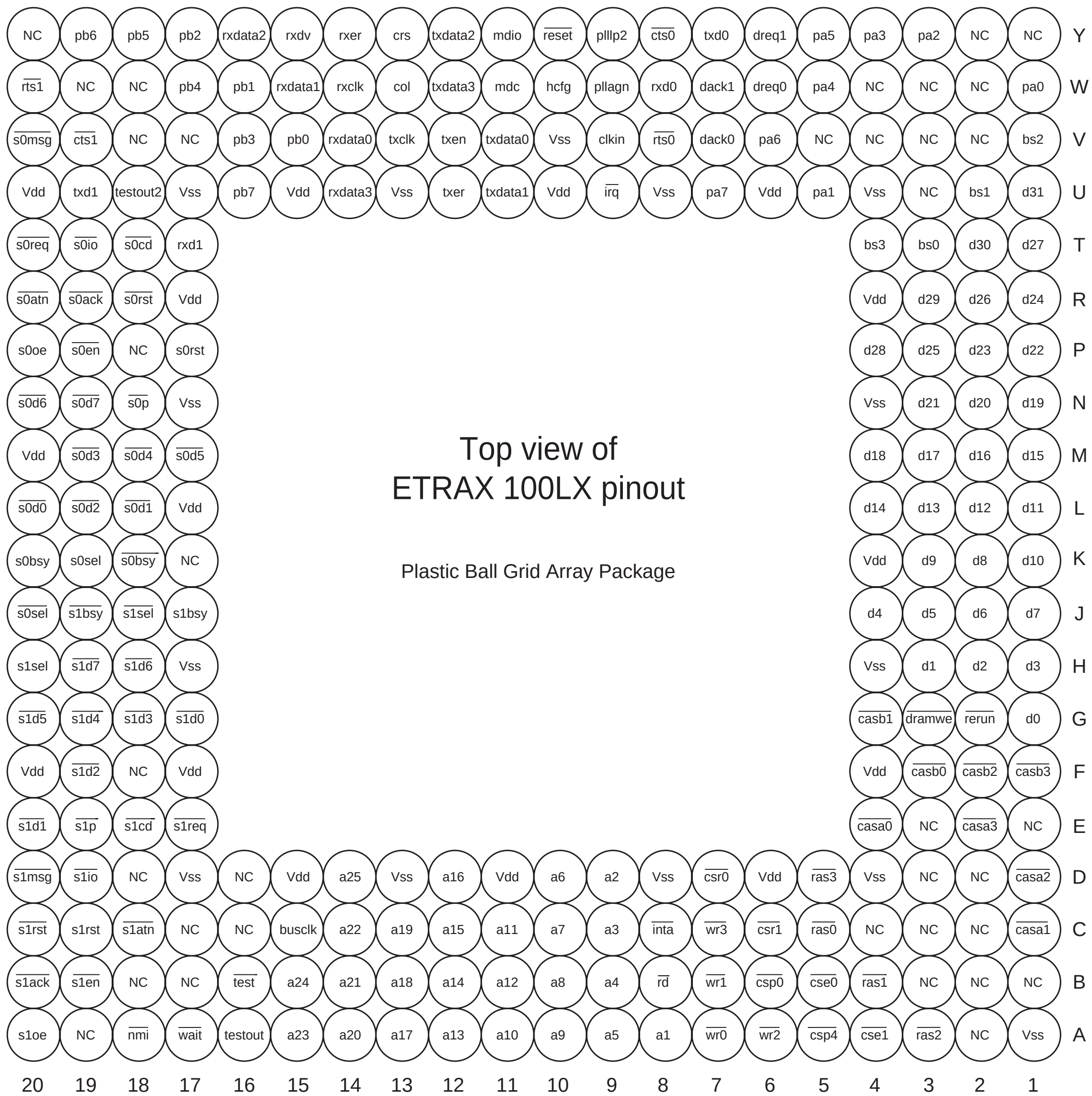


Figure 19-1 The ETRAX 100LX Pinout

19.2 Clock and PLL Signals

Solder Ball	Name	Direction	Description
V9	clkin	in	External clock input.
W9	pllagn	out	PLL loop filter internal ground connection.
Y9	plllp2	in/out	PLL loop filter.

Table 19-1 Clock and Phase-locked Loop Signals

19.3 Power and Ground Signals

Solder Ball	Name	Description
A1 D4 D8 D13 D17 H4 H17 N4 N17 U4 U8 U13 U17 V10	V_{ss}	Ground connection.
D6 D11 D15 F4 F17 F20 K4 L17 M20 R4 R17 U6 U10 U15 U20	V_{dd}	Supply voltage, 3.3 V.

Table 19-2 Power and Ground Signals

19.4 Bus Interface Signals

Data Bus

Solder Ball	Direction	Pin Name
G1	in/out	d0
H3	in/out	d1
H2	in/out	d2
H1	in/out	d3
J4	in/out	d4
J3	in/out	d5
J2	in/out	d6
J1	in/out	d7
K2	in/out	d8
K3	in/out	d9
K1	in/out	d10
L1	in/out	d11
L2	in/out	d12
L3	in/out	d13
L4	in/out	d14
M1	in/out	d15
M2	in/out	d16
M3	in/out	d17
M4	in/out	d18
N1	in/out	d19
N2	in/out	d20
N3	in/out	d21
P1	in/out	d22
P2	in/out	d23
R1	in/out	d24
P3	in/out	d25
R2	in/out	d26
T1	in/out	d27
P4	in/out	d28
R3	in/out	d29
T2	in/out	d30
U1	in/out	d31

Address Bus

Solder Ball	Direction	Pin Name
A8	out	a1 (Note 1)
D9	out	a2
C9	out	a3
B9	out	a4
A9	out	a5
D10	out	a6
C10	out	a7
B10	out	a8
A10	out	a9
A11	out	a10
C11	out	a11
B11	out	a12
A12	out	a13
B12	out	a14
C12	out	a15
D12	out	a16
A13	out	a17
B13	out	a18
C13	out	a19
A14	out	a20
B14	out	a21
C14	out	a22
A15	out	a23 (Note 2)
B15	out	a24 (Note 2)
D14	out	a25 (Note 2)

Table 19-3 Data and Address Buses

Note 1: Solder ball A8 is dual-function. See table 19-5.

Note 2: Solder balls A15, B15 and D14 are dual-function. See table 19-6.

Chip Select Signals

Solder Ball	Direction	Pin Name	Description
B5	out	$\overline{\text{cse0}}$	Chip select signal for EPROM/flashPROM 0.
A4	out	$\overline{\text{cse1}}$	Chip select signal for EPROM/flashPROM 1.
D7	out	$\overline{\text{csr0}}$	Chip select signal for SRAM 0.
C6	out	$\overline{\text{csr1}}$	Chip select signal for SRAM 1.
B6	out	$\overline{\text{csp0}}$	Peripheral chip select signal 0.
A5	out	$\overline{\text{csp4}}$	Peripheral chip select signal 4.

Table 19-4 Chip Select Signals

Peripheral chip select signals $\overline{\text{csp1}}$ to $\overline{\text{csp3}}$ and $\overline{\text{csp5}}$ to $\overline{\text{csp7}}$ are multiplexed with bits pb2 to pb7 in general port PB. See tables 19-19 and 19-20.

Read/Write Strobes

Solder Ball	Direction	Bytewise Write Enable	Common Write Enable		Description
			16-bit Mode	32-bit Mode	
B8	out	$\overline{\text{rd}}$	$\overline{\text{rd}}$	$\overline{\text{rd}}$	Read strobe, common to all four bytes of the data bus. Not active during DRAM access.
A7	out	$\overline{\text{wr0}}$			Write strobe for byte 0 of the data bus.
			$\overline{\text{be0}}$	$\overline{\text{be0}}$	Byte enable strobe for byte 0 of the data bus.
B7	out	$\overline{\text{wr1}}$			Write strobe for byte 1 of the data bus.
			$\overline{\text{be1}}$	$\overline{\text{be1}}$	Byte enable strobe for byte 1 of the data bus.
A6	out	$\overline{\text{wr2}}$			Write strobe for byte 2 of the data bus.
				$\overline{\text{be2}}$	Byte enable strobe for byte 2 of the data bus.
C7	out	$\overline{\text{wr3}}$			Write strobe for byte 3 of the data bus.
			$\overline{\text{we}}$	$\overline{\text{we}}$	Write enable strobe, common to all four bytes of the data bus.
A8	out	a1	a1		Least significant address bit.
				$\overline{\text{be3}}$	Byte enable strobe for byte 3 of the data bus.

Table 19-5 Read/Write Strobe Signals for Bytewise and Common Write Enable Modes

Asynchronous and Synchronous DRAM Signals

		Asynchronous DRAM			
Solder Ball	Direction	Bytewise Mode	Bankwise Mode	Sync. DRAM	Description
E4	out	$\overline{casa0}$			Column address strobe for byte 0 in Async. DRAM bank 0 and 1.
			$\overline{casa0}$		Column address strobe for Async. DRAM bank 0.
				dqm0	Data qualify mask 0.
C1	out	$\overline{casa1}$			Column address strobe for byte 1 in Async. DRAM bank 0 and 1.
			$\overline{casa1}$		Column address strobe for Async. DRAM bank 1.
				dqm1	Data qualify mask 1.
D1	out	$\overline{casa2}$			Column address strobe for byte 2 in Async. DRAM bank 0 and 1.
			$\overline{casa2}$		Column address strobe for Async. DRAM bank 2.
				dqm2	Data qualify mask 2.
E2	out	$\overline{casa3}$			Column address strobe for byte 3 in Async. DRAM bank 0 and 1.
			$\overline{casa3}$		Column address strobe for Async. DRAM bank 3.
				dqm3	Data qualify mask 3.
F3	out	$\overline{casb0}$			Column address strobe for byte 0 in Async. DRAM bank 2 and 3.
			$\overline{be0}$		Enable signal for byte 0 of the data bus.
				dqm4	Data qualify mask 4.
G4	out	$\overline{casb1}$			Column address strobe for byte 1 in Async. DRAM bank 2 and 3.
			$\overline{be1}$		Enable signal for byte 1 of the data bus.
				dqm5	Data qualify mask 5.
F2	out	$\overline{casb2}$			Column address strobe for byte 2 in Async. DRAM bank 2 and 3.
			$\overline{be2}$		Enable signal for byte 2 of the data bus.
				dqm6	Data qualify mask 6.
F1	out	$\overline{casb3}$			Column address strobe for byte 3 in Async. DRAM bank 2 and 3.
			$\overline{be3}$		Enable signal for byte 2 of the data bus.
				dqm7	Data qualify mask 7.
G3	out	$\overline{dramwe}$	$\overline{dramwe}$		Write enable signal for Async. DRAM.
	in/out			dqs	DDR data qualify strobe.
C5	out	$\overline{ras0}$	$\overline{ras0}$		Row address strobe for Async. DRAM bank 0.
				$\overline{csd0}$	Chip select signal for Sync. DRAM group 0.
B4	out	$\overline{ras1}$	$\overline{ras1}$		Row address strobe for Async. DRAM bank 1.
				$\overline{csd1}$	Chip select signal for Sync. DRAM group 1.
A3	out	$\overline{ras2}$	$\overline{ras2}$		Row address strobe for Async. DRAM bank 2.
				clk	Master clock signal for Sync. DRAM.
D5	out	$\overline{ras3}$	$\overline{ras3}$		Row address strobe for Async. DRAM bank 3.
				cke	Clock enable for Sync. DRAM.
A15	out	a23	a23		Bit 23 of address bus.
				$\overline{sdram\_we}$	Write enable signal for Sync. DRAM.
B15	out	a24	a24		Bit 24 of address bus.
				$\overline{sdram\_cas}$	Column address strobe for Sync. DRAM.
D14	out	a25	a25		Bit 25 of address bus.
				$\overline{sdram\_ras}$	Row address strobe for Sync. DRAM.

Table 19-6 Asynchronous and Synchronous DRAM Signals

Miscellaneous Bus Interface Signals

Solder Ball	Direction	Name	Description
G2	in	$\overline{\text{rerun}}$	Bus rerun signal.
W6	in	dreq0	DMA request, external DMA0.
Y6	in	dreq1	DMA request, external DMA1.
V7	out	dack0	DMA acknowledge, external DMA0.
W7	out	dack1	DMA acknowledge, external DMA1.
U9	in	$\overline{\text{irq}}$	Interrupt request.
A18	in	$\overline{\text{nmi}}$	Non maskable interrupt request.
A17	in	$\overline{\text{wait}}$	External wait state input.
C8	out	$\overline{\text{inta}}$	External interrupt acknowledge.
Y10	in	$\overline{\text{reset}}$	System reset.
W10	in	hcfg	Hardware configuration for serial ports 2 and 3.

Table 19-7 Miscellaneous Bus Interface Signals

19.5 Logic Analyzer Mode and Test Signals

Solder Ball	Direction	Name	Description
T3	in/out	bs0	Bus status, bit 0 (Note 3).
U2	in/out	bs1	Bus status, bit 1 (Note 3).
V1	in/out	bs2	Bus status, bit 2 (Note 3).
T4	in/out	bs3	Bus status, bit 3 (Note 3).
B16	in	$\overline{\text{test}}$	Test input, should be high for normal operation.
A16	out	testout	Test output, must not be connected.
C15	out	busclk	Bus synchronization clock. Used for debug purposes.
U18	out	testout2	Test output, must not be connected.

Table 19-8 Logic Analyzer Mode and Test Signals

Note 3: These signals are for bus configuration during power-on reset. They are used as status outputs for debugging purposes when reset is inactive.

19.6 General Port PA Signals

Solder Ball	Direction	Name	Description
W1	in/out	pa0	General port PA, bit 0.
U5	in/out	pa1	General port PA, bit 1.
Y3	in/out	pa2	General port PA, bit 2.
Y4	in/out	pa3	General port PA, bit 3.
W5	in/out	pa4	General port PA, bit 4.
Y5	in/out	pa5	General port PA, bit 5.
V6	in/out	pa6	General port PA, bit 6.
U7	in/out	pa7	General port PA, bit 7.

Table 19-9 General Port PA Signals

19.7 Asynchronous Serial Port 0 Signals

Solder Ball	Direction	Name	Description
Y7	out	txd0	Transmit data, serial port 0.
V8	out	$\overline{\text{rts0}}$	Request to send, serial port 0.
W8	in	rxd0	Receive data, serial port 0.
Y8	in	$\overline{\text{cts0}}$	Clear to send, serial port 0.

Table 19-10 Serial Port 0 Signals

Note 4: I/O signals at Asynchronous Serial Ports p1 to p3 are multiplexed on to pins used by other interfaces. Sections 19-9 and 19-10 refer.

19.8 Network Interface Signals

Solder Ball	Direction	Name	MII Usage	SNI Usage
Y11	in/out	mdio	Management data.	General I/O.
W11	out	mdc	Management clock.	General output.
V11	out	txdata0	Data out, bit 0.	Data out.
U11	out	txdata1	Data out, bit 1.	General output.
Y12	out	txdata2	Data out, bit 2.	General output.
W12	out	txdata3	Data out, bit 3.	General output.
V12	out	txen	Transmit enable.	Transmit enable.
U12	out	txer	Transmit error/ 25 MHz clock/ Address recognized.	General output.
Y13	in	crs	Carrier sense.	Carrier sense.
W13	in	col	Collision.	Collision.
V13	in	txclk	Transmit clock.	Transmit clock.
Y14	in	rxer	Receive error.	General input.
W14	in	rxclk	Receive clock.	Receive clock.
Y15	in	rxdv	Data in valid.	Not used.
V14	in	rxdata0	Data in, bit 0.	Data in.
W15	in	rxdata1	Data in, bit 1.	General input.
Y16	in	rxdata2	Data in, bit 2.	General input.
U14	in	rxdata3	Data in, bit 3.	General input.

Table 19-11 Network Interface Signals

19.9 Multiplexed Signal Groups

To optimize chip efficiency and minimize the device footprint, certain interfaces share a number of I/O pins. The input and outputs to and from these interfaces are multiplexed on to these common pins. The table below lists the interfaces whose inputs and outputs are multiplexed in this way.

Interface
Asynchronous Serial Port p1 Asynchronous Serial Port p2 Asynchronous Serial Port p3
Synchronous Serial Port p1 Synchronous Serial Port p3
Shared RAM (8-bit) Shared RAM-W (16-bit wide)
Parallel Port p0 Parallel Port p1 Parallel Port-W (16-bit wide)
SCSI-8 Port p0 SCSI-8 Port p1 SCSI-W (16-bit wide)
ATA interface
Additional Chip Select (CSP) I2C Port
USB interface port p1 USB interface port p2
General I/O pins

Table 19-12 Multiplexed Interfaces

The I/O pins on to which the interface signals are multiplexed are arranged in six groups denoted A to F respectively. The table on the next page maps the relationships between the groups of pins and the interfaces that use them. It shows that some interfaces are mutually exclusive - they cannot use the I/O pins simultaneously. For example it is not possible to use SCSI-8 p0 at the same time as Asynchronous Serial Port p2 because the four Group B pins used by the serial port are also required for the SCSI interface.

Pins that are not used by a particular interface are available for general I/O purposes.

I/O PIN GROUPS					
A (19 pins)	**B (4 pins)**	**C (4 pins)**	**D (19 pins)**	**E (4 pins)**	**F (8 pins)**
–	Asynchronous Serial Port p2	Asynchronous Serial Port p3	–	Asynchronous Serial Port p1	–
–	–	Synchronous Serial Port p3	–	Synchronous Serial Port p1	Synchronous Serial Port p1 (Note 5)
–	–	–	–	–	Synchronous Serial Port p3 (Note 5)
Shared RAM	–	–	–	–	–
Shared RAM-W	–	–	Shared RAM-W	–	–
Parallel Port p0	–	–	Parallel Port p1	–	–
Parallel Port-W	–	–	Parallel Port-W	–	–
SCSI-8 Port p0	SCSI-8 Port p0	–	–	–	SCSI-8 Port p0 (Note 5)
–	–	SCSI-8 Port p1	SCSI-8 Port p1	–	SCSI-8 Port p1 (Note 5)
SCSI-W Port	SCSI-W Port	–	SCSI-W Port	–	SCSI-W Port
ATA Port	ATA Port	ATA Port	ATA Port	–	–
–	–	–	–	–	CSP and I2C Port
–	–	–	USB Port p2	USB Port p1	USB Port p1
General I/O Port	General I/O Port	General I/O Port	General I/O Port	General I/O Port	General I/O Port

Table 19-13 Multiplexed Interfaces and I/O Pin Groups

Note 5: These ports use only two pins of I/O pin Group F. Consequently the two synchronous serial ports are not mutually exclusive of each other because unused pins remain available. Similarly the two SCSI-8 ports are not mutually exclusive of each other. However the *pairs of ports* are mutually exclusive because, in Group F, Synchronous Serial Port p1 and SCSI-8 p0 both use pin W17, and Synchronous Serial Port p3 and SCSI-8 p1 both use pin U16.

19.9.1 Multiplexed I/O Signals - Group A

	INTERFACES															
Pin	SCSI-8 p0		SCSI-W		ATA		Parallel Port p0		Parallel Port-W		Shared RAM/ Shared RAM-W				General I/O	
V20	$\overline{\text{s0msg}}$	in/ out	$\overline{\text{s0msg}}$	in/ out	iordy	in	p0perror	in	p0perror	in	pr_adr0	in	-	-	g5	in
T18	$\overline{\text{s0cd}}$	in/ out	$\overline{\text{s0cd}}$	in/ out	dmarq0	in	$\overline{\text{p0ack}}$	in	$\overline{\text{p0ack}}$	in	pr_adr1	in	-	-	g4	in
T19	$\overline{\text{s0io}}$	in/ out	$\overline{\text{s0io}}$	in/ out	dmarq1	in	p0busy	in	p0busy	in	$\overline{\text{intio}}$	in	-	-	g3	in
T20	$\overline{\text{s0req}}$	in	$\overline{\text{s0req}}$	in	dmarq2	in	$\overline{\text{p0fault}}$	in	$\overline{\text{p0fault}}$	in	rd_wr	in	-	-	g2	in
R18	$\overline{\text{s0rst}}$	in	$\overline{\text{s0rst}}$	in	dmarq3	in	p0select	in	p0select	in	$\overline{\text{pr\_req}}$	in	-	-	g1	in
P17	s0rst	out	s0rst	out	$\overline{\text{cs0}}$	out	p0data_oe	out	p0data_oe	out	$\overline{\text{pr\_int}}$	out	-	-	g5	out
R19	$\overline{\text{s0ack}}$	out	$\overline{\text{s0ack}}$	out	$\overline{\text{cs1}}$	out	$\overline{\text{p0selectin}}$	out	$\overline{\text{p0selectin}}$	out	$\overline{\text{pr\_ack}}$	out	-	-	g4	out
R20	$\overline{\text{s0atn}}$	out	$\overline{\text{s0atn}}$	out	a0	out	$\overline{\text{p0autofd}}$	out	$\overline{\text{p0autofd}}$	out	a_sel	out	-	-	g3	out
K20	s0bsy	out	s0bsy	out	a1	out	$\overline{\text{p0strobe}}$	out	$\overline{\text{p0strobe}}$	out	-	-	-	-	g2	out
P20	s0oe	out	s0oe	out	a2	out	$\overline{\text{p0init}}$	out	$\overline{\text{p0init}}$	out	-	-	-	-	g1	out
N18	$\overline{\text{s0p}}$	in/ out	$\overline{\text{s0p}}$	in/ out	$\overline{\text{dmack0}}$	out	-	-	-	-	-	-	-	-	g0	in/ out
N19	$\overline{\text{s0d7}}$	in/ out	$\overline{\text{s0d7}}$	in/ out	d7	in/ out	p0d7	in/ out	p0d7	in/ out	pr_d7	in/ out	-	-	g15	in/ out
N20	$\overline{\text{s0d6}}$	in/ out	$\overline{\text{s0d6}}$	in/ out	d6	in/ out	p0d6	in/ out	p0d6	in/ out	pr_d6	in/ out	-	-	g14	in/ out
M17	$\overline{\text{s0d5}}$	in/ out	$\overline{\text{s0d5}}$	in/ out	d5	in/ out	p0d5	in/ out	p0d5	in/ out	pr_d5	in/ out	-	-	g13	in/ out
M18	$\overline{\text{s0d4}}$	in/ out	$\overline{\text{s0d4}}$	in/ out	d4	in/ out	p0d4	in/ out	p0d4	in/ out	pr_d4	in/ out	-	-	g12	in/ out
M19	$\overline{\text{s0d3}}$	in/ out	$\overline{\text{s0d3}}$	in/ out	d3	in/ out	p0d3	in/ out	p0d3	in/ out	pr_d3	in/ out	-	-	g11	in/ out
L19	$\overline{\text{s0d2}}$	in/ out	$\overline{\text{s0d2}}$	in/ out	d2	in/ out	p0d2	in/ out	p0d2	in/ out	pr_d2	in/ out	-	-	g10	in/ out
L18	$\overline{\text{s0d1}}$	in/ out	$\overline{\text{s0d1}}$	in/ out	d1	in/ out	p0d1	in/ out	p0d1	in/ out	pr_d1	in/ out	-	-	g9	in/ out
L20	$\overline{\text{s0d0}}$	in/ out	$\overline{\text{s0d0}}$	in/ out	d0	in/ out	p0d0	in/ out	p0d0	in/ out	pr_d0	in/ out	-	-	g8	in/ out

Table 19-14 Multiplexed I/O Signals - Group A

19.9.2 Multiplexed I/O Signals - Group B

	INTERFACES															
Pin	SCSI-8 p0		SCSI-W		ATA		Async. Serial Port p2								General I/O	
P19	$\overline{\text{s0en}}$	out	$\overline{\text{s0en}}$	out	$\overline{\text{dior0}}$	out	$\overline{\text{rts2}}$	out	-	-	-	-	-	-	g7	out
K19	s0sel	out	s0sel	out	$\overline{\text{dior1}}$	out	txd2	out	-	-	-	-	-	-	g6	out
K18	$\overline{\text{s0bsy}}$	in	$\overline{\text{s0bsy}}$	in	intrq0	in	$\overline{\text{cts2}}$	in	-	-	-	-	-	-	g7	in
J20	$\overline{\text{s0sel}}$	in	$\overline{\text{s0sel}}$	in	intrq1	in	rxd2	in	-	-	-	-	-	-	g6	in

Table 19-15 Multiplexed I/O Signals - Group B

19.9.3 Multiplexed I/O Signals - Group C

	INTERFACES															
Pin	**SCSI-8 p1**				**ATA**		**Async. Serial Port p3**		**Sync. Serial Port p3**						**General I/O**	
J19	$\overline{s1bsy}$	in	-	-	intrq2	in	$\overline{cts3}$	in	ss3_in2	in	-	-	-	-	g31	in
J18	$\overline{s1sel}$	in	-	-	intrq3	in	rxd3	in	ss3_in1	in	-	-	-	-	g30	in
B19	$\overline{s1en}$	out	-	-	$\overline{dior2}$	out	$\overline{rts3}$	out	ss3_out2	out	-	-	-	-	g31	out
H20	s1sel	out	-	-	$\overline{dior3}$	out	txd3	out	ss3_out1	out	-	-	-	-	g30	out

Table 19-16 Multiplexed I/O Signals - Group C

19.9.4 Multiplexed I/O Signals - Group D

	INTERFACES															
Pin	**SCSI-8 p1**		**SCSI-W**		**ATA**		**Parallel Port p1**		**Parallel Port-W**		**Shared RAM-W**		**USB Port p2**		**General I/O**	
H19	$\overline{s1d7}$	in/out	$\overline{s0d15}$	in/out	d15	in/out	p1d7	in/out	p0d15	in/out	pr_d15	in/out	-	-	g23	in/out
H18	$\overline{s1d6}$	in/out	$\overline{s0d14}$	in/out	d14	in/out	p1d6	in/out	p0d14	in/out	pr_d14	in/out	-	-	g22	in/out
G20	$\overline{s1d5}$	in/out	$\overline{s0d13}$	in/out	d13	in/out	p1d5	in/out	p0d13	in/out	pr_d13	in/out	-	-	g21	in/out
G19	$\overline{s1d4}$	in/out	$\overline{s0d12}$	in/out	d12	in/out	p1d4	in/out	p0d12	in/out	pr_d12	in/out	-	-	g20	in/out
G18	$\overline{s1d3}$	in/out	$\overline{s0d11}$	in/out	d11	in/out	p1d3	in/out	p0d11	in/out	pr_d11	in/out	-	-	g19	in/out
F19	$\overline{s1d2}$	in/out	$\overline{s0d10}$	in/out	d10	in/out	p1d2	in/out	p0d10	in/out	pr_d10	in/out	-	-	g18	in/out
E20	$\overline{s1d1}$	in/out	$\overline{s0d9}$	in/out	d9	in/out	p1d1	in/out	p0d9	in/out	pr_d9	in/out	-	-	g17	in/out
G17	$\overline{s1d0}$	in/out	$\overline{s0d8}$	in/out	d8	in/out	p1d0	in/out	p0d8	in/out	pr_d8	in/out	-	-	g16	in/out
E19	$\overline{s1p}$	in/out	$\overline{s0p1}$	in/out	$\overline{dmack1}$	out	-	-	-	-	-	-	-	-	g24	in/out
D20	$\overline{s1msg}$	in/out	-	-	$\overline{dmack2}$	out	p1perror	in	-	-	-	-	$\overline{usb2\_oe}$	out	g29	in
E18	$\overline{s1cd}$	in/out	-	-	$\overline{dmack3}$	out	$\overline{p1ack}$	in	-	-	-	-	usb2_speed	out	g28	in
D19	$\overline{s1io}$	in/out	-	-	g27	out	p1busy	in	-	-	-	-	usb2_rcv	in	g27	in
E17	$\overline{s1req}$	in	-	-	-	-	$\overline{p1fault}$	in	-	-	-	-	usb2_vp	in	g26	in
C20	$\overline{s1rst}$	in	-	-	-	-	p1select	in	-	-	-	-	usb2_vm	in	g25	in
C19	s1rst	out	-	-	$\overline{diow0}$	out	p1data_oe	out	-	-	-	-	-	-	g29	out
B20	$\overline{s1ack}$	out	-	-	$\overline{diow1}$	out	$\overline{p1selectin}$	out	-	-	-	-	-	-	g28	out
C18	$\overline{s1atn}$	out	-	-	$\overline{diow2}$	out	$\overline{p1autofd}$	out	-	-	-	-	usb2_vpo	out	g27	out
J17	s1bsy	out	$\overline{s0enhiid}$	out	$\overline{diow3}$	out	$\overline{p1strobe}$	out	-	-	-	-	usb2_vmo	out	g26	out
A20	s1oe	out	s1oe	out	ext_oe	out	$\overline{p1init}$	out	-	-	-	-	-	-	g25	out

Table 19-17 Multiplexed I/O Signals - Group D

19.9.5 Multiplexed I/O Signals - Group E

	INTERFACES															
Pin							Sync. Serial Port p1		Async. Serial Port p1				USB Port p1			
U19	-	-	-	-	-	-	ss1_out1	out	txd1	out	-	-	usb1_oe	out		
W20	-	-	-	-	-	-	ss1_out2	out	rts1	out	-	-	usb1_speed	out		
T17	-	-	-	-	-	-	ss1_in1	in	rxd1	in	-	-	usb1_rcv	in		
V19	-	-	-	-	-	-	ss1_in2	in	cts1	in	-	-	usb1_vp	in		

Table 19-18 Multiplexed I/O Signals - Group E

19.9.6 Multiplexed I/O Signals - Group F

	INTERFACES															
Pin	SCSI-8 p0		SCSI-8 p1		SCSI-W		Sync. Serial Port p1		Sync. Serial Port p3		CSP and I2C		USB Port p1		General I/O	
V15	-	-	-	-	-	-	-	-	-	-	i2c_d	in/out	-	-	pb0	in/out
W16	-	-	-	-	-	-	-	-	-	-	i2c_clk	out	-	-	pb1	in/out
Y17	-	-	-	-	-	-	-	-	-	-	csp1	out	usb1_vpo	out	pb2	in/out
V16	-	-	-	-	-	-	-	-	-	-	csp2	out	usb1_vmo	out	pb3	in/out
W17	s0enph	out	-	-	s0enph	out	ss1_io3	in/out	-	-	csp3	out	-	-	pb4	in/out
Y18	-	-	-	-	-	-	-	-	-	-	csp5	out	usb1_vm	in	pb5	in/out
Y19	-	-	-	-	-	-	-	-	-	-	csp6	out	-	-	pb6	in/out
U16	-	-	s1enph	out	s0enloid	out	-	-	ss3_io3	in/out	csp7	out	-	-	pb7	in/out

Table 19-19 Multiplexed I/O Signals - Group F

19.10 Multiplexed Interfaces

This section provides signal assignment details of the various interfaces that are multiplexed on to the I/O pins.

19.10.1 SCSI Ports

SCSI-8 Port p0

The pins listed below are used by SCSI-8 Port p0 in 8-bit mode. These pins are also used by the SCSI-W port in 16-bit (wide) mode (see table 19-20).

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\text{s0msg}}$	$\overline{\text{s0msg}}$	in/out (Note 6)	Message, driven by target during message phase.
T18	$\overline{\text{s0cd}}$	$\overline{\text{s0cd}}$	in/out (Note 6)	Control/Data, driven by target to determine whether control or data information is on the bus.
T19	$\overline{\text{s0io}}$	$\overline{\text{s0io}}$	in/out (Note 6)	Input/Output, driven by target and indicates bus direction.
R19	$\overline{\text{s0ack}}$	$\overline{\text{s0ack}}$	out	Acknowledgement signal in the information transfer handshake.
T20	$\overline{\text{s0req}}$	$\overline{\text{s0req}}$	in	Request signal in the information transfer handshake.
R20	$\overline{\text{s0atn}}$	$\overline{\text{s0atn}}$	out	Attention condition indicator from ETRAX 100LX to target.
K19	s0sel	s0sel	out	Select or reselect signal from SCSI interface to SCSI buffer.
J20	$\overline{\text{s0sel}}$	$\overline{\text{s0sel}}$	in (Note 7)	-SEL signal from SCSI bus to SCSI interface.
K20	s0bsy	s0bsy	out	Bus in use signal from SCSI interface to SCSI buffer.
K18	$\overline{\text{s0bsy}}$	$\overline{\text{s0bsy}}$	in (Note 7)	-BSY signal from SCSI bus to SCSI interface.
P17	s0rst	s0rst	out	Bus reset signal from SCSI interface to SCSI buffer.
R18	$\overline{\text{s0rst}}$	$\overline{\text{s0rst}}$	in (Note 7)	-RST signal from SCSI bus to SCSI interface.
N18	$\overline{\text{s0p}}$	$\overline{\text{s0p}}$	in/out	Data bus parity.
L20 L18 L19 M19 M18 M17 N20 N19	$\overline{\text{s0d0}}$ $\overline{\text{s0d1}}$ $\overline{\text{s0d2}}$ $\overline{\text{s0d3}}$ $\overline{\text{s0d4}}$ $\overline{\text{s0d5}}$ $\overline{\text{s0d6}}$ $\overline{\text{s0d7}}$	$\overline{\text{s0d0}}$ $\overline{\text{s0d1}}$ $\overline{\text{s0d2}}$ $\overline{\text{s0d3}}$ $\overline{\text{s0d4}}$ $\overline{\text{s0d5}}$ $\overline{\text{s0d6}}$ $\overline{\text{s0d7}}$	in/out in/out in/out in/out in/out in/out in/out in/out	Data bus of port p0 in SCSI 8-bit mode.
P20	s0oe	s0oe	out	External bus driver direction for signals $\overline{\text{s0d0}}$ to $\overline{\text{s0d7}}$ and $\overline{\text{s0p}}$.
P19	$\overline{\text{s0en}}$	$\overline{\text{s0en}}$	out	External driver output enable for signals $\overline{\text{s0ack}}$ and $\overline{\text{s0atn}}$.
W17	pb4	$\overline{\text{s0enph}}$	in/out	SCSI-0 phase enable for software ID select.

Table 19-20 SCSI-8 Port p0 Signals

For Notes 6 and 7, refer to *SCSI-W Port* on page 566.

SCSI-8 Port p1

The pins listed below are used by SCSI-8 Port p1 in 8-bit mode. Some pins are also used by the SCSI-W Port in 16-bit (wide) mode (see tables 19-20, 19-21, and 19-22).

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
D20	$\overline{\text{s1msg}}$	$\overline{\text{s1msg}}$	in/out (Note 6)	Message, driven by target during message phase.
E18	$\overline{\text{s1cd}}$	$\overline{\text{s1cd}}$	in/out (Note 6)	Control/Data, driven by target to determine whether control or data information is on the bus.
D19	$\overline{\text{s1io}}$	$\overline{\text{s1io}}$	in/out (Note 6)	Input/Output, driven by target and indicates bus direction.
B20	$\overline{\text{s1ack}}$	$\overline{\text{s1ack}}$	out	Acknowledgement signal in the information transfer handshake.
E17	$\overline{\text{s1req}}$	$\overline{\text{s1req}}$	in	Request signal in the information transfer handshake.
C18	$\overline{\text{s1atn}}$	$\overline{\text{s1atn}}$	out	Attention condition indicator from ETRAX 100LX to target.
H20	s1sel	s1sel	out	Select or reselect signal from SCSI interface to SCSI buffer.
J18	$\overline{\text{s1sel}}$	$\overline{\text{s1sel}}$	in (Note 7)	-SEL signal from SCSI bus to SCSI interface.
J17	s1bsy	s1bsy	out	Bus in use signal from SCSI interface to SCSI buffer.
J19	$\overline{\text{s1bsy}}$	$\overline{\text{s1bsy}}$	in (Note 7)	-BSY signal from SCSI bus to SCSI interface.
C19	s1rst	s1rst	out	Bus reset signal from SCSI interface to SCSI buffer.
C20	$\overline{\text{s1rst}}$	$\overline{\text{s1rst}}$	in (Note 7)	-RST signal from SCSI bus to SCSI interface.
E19	$\overline{\text{s1p}}$	$\overline{\text{s1p}}$	in/out	Data bus parity.
G17 E20 F19 G18 G19 G20 H18 H19	$\overline{\text{s1d0}}$ $\overline{\text{s1d1}}$ $\overline{\text{s1d2}}$ $\overline{\text{s1d3}}$ $\overline{\text{s1d4}}$ $\overline{\text{s1d5}}$ $\overline{\text{s1d6}}$ $\overline{\text{s1d7}}$	$\overline{\text{s1d0}}$ $\overline{\text{s1d1}}$ $\overline{\text{s1d2}}$ $\overline{\text{s1d3}}$ $\overline{\text{s1d4}}$ $\overline{\text{s1d5}}$ $\overline{\text{s1d6}}$ $\overline{\text{s1d7}}$	in/out in/out in/out in/out in/out in/out in/out in/out	Data bus of port p1 in SCSI 8-bit mode.
A20	s1oe	s1oe	out	External bus driver direction for signals $\overline{\text{s1d0}}$ to $\overline{\text{s1d7}}$ and $\overline{\text{s1p}}$.
B19	$\overline{\text{s1en}}$	$\overline{\text{s1en}}$	out	External driver output enable for signals $\overline{\text{s1ack}}$ and $\overline{\text{s1atn}}$.
U16	pb7	$\overline{\text{s1enph}}$	in/out	SCSI-1 phase enable for software ID select.

Table 19-21 SCSI-8 Port p1 Signals

For Notes 6 and 7, refer to *SCSI-W Port* on page 566.

SCSI-W Port

In 16-bit mode the SCSI-W Port uses the following I/O pins, which are also used by SCSI-8 Ports p0 and p1 in 8-bit mode.

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\mathrm{s0msg}}$	$\overline{\mathrm{s0msg}}$	in/out (Note 6)	Message, driven by target during message phase.
T18	$\overline{\mathrm{s0cd}}$	$\overline{\mathrm{s0cd}}$	in/out (Note 6)	Control/Data, driven by target to determine whether control or data information is on the bus.
T19	$\overline{\mathrm{s0io}}$	$\overline{\mathrm{s0io}}$	in/out (Note 6)	Input/output driven by target to show bus direction.
R19	$\overline{\mathrm{s0ack}}$	$\overline{\mathrm{s0ack}}$	out	Acknowledge signal in transfer handshake.
T20	$\overline{\mathrm{s0req}}$	$\overline{\mathrm{s0req}}$	in	Request signal in the information transfer handshake.
R20	$\overline{\mathrm{s0atn}}$	$\overline{\mathrm{s0atn}}$	out	Attention condition indicator from ETRAX 100LX to target.
K19	s0sel	s0sel	out	Select or reselect signal from SCSI interface to SCSI buffer.
J20	$\overline{\mathrm{s0sel}}$	$\overline{\mathrm{s0sel}}$	in (Note 7)	-SEL signal from SCSI bus to SCSI interface.
K20	s0bsy	s0bsy	out	Bus in use signal from SCSI interface to SCSI buffer.
J17	s1bsy	$\overline{\mathrm{s0enhiid}}$	out	Enable arbitration ID 7 - 0 for software ID.
K18	$\overline{\mathrm{s0bsy}}$	$\overline{\mathrm{s0bsy}}$	in (Note 7)	-BSY signal from SCSI bus to SCSI interface.
P17	s0rst	s0rst	out	Bus reset signal from SCSI interface to SCSI buffer.
R18	$\overline{\mathrm{s0rst}}$	$\overline{\mathrm{s0rst}}$	in (Note 7)	-RST signal from SCSI bus to SCSI interface.
N18	$\overline{\mathrm{s0p}}$	$\overline{\mathrm{s0p}}$	in/out	Data bus low byte parity.
E19	$\overline{\mathrm{s1p}}$	$\overline{\mathrm{s0p1}}$	in/out	Data bus high byte parity.
L20 L18 L19 M19 M18 M17 N20 N19	$\overline{\mathrm{s0d0}}$ $\overline{\mathrm{s0d1}}$ $\overline{\mathrm{s0d2}}$ $\overline{\mathrm{s0d3}}$ $\overline{\mathrm{s0d4}}$ $\overline{\mathrm{s0d5}}$ $\overline{\mathrm{s0d6}}$ $\overline{\mathrm{s0d7}}$	$\overline{\mathrm{s0d0}}$ $\overline{\mathrm{s0d1}}$ $\overline{\mathrm{s0d2}}$ $\overline{\mathrm{s0d3}}$ $\overline{\mathrm{s0d4}}$ $\overline{\mathrm{s0d5}}$ $\overline{\mathrm{s0d6}}$ $\overline{\mathrm{s0d7}}$	in/out in/out in/out in/out in/out in/out in/out in/out	Low byte of data bus in SCSI-W mode.
G17 E20 F19 G18 G19 G20 H18 H19	$\overline{\mathrm{s1d0}}$ $\overline{\mathrm{s1d1}}$ $\overline{\mathrm{s1d2}}$ $\overline{\mathrm{s1d3}}$ $\overline{\mathrm{s1d4}}$ $\overline{\mathrm{s1d5}}$ $\overline{\mathrm{s1d6}}$ $\overline{\mathrm{s1d7}}$	$\overline{\mathrm{s0d8}}$ $\overline{\mathrm{s0d9}}$ $\overline{\mathrm{s0d10}}$ $\overline{\mathrm{s0d11}}$ $\overline{\mathrm{s0d12}}$ $\overline{\mathrm{s0d13}}$ $\overline{\mathrm{s0d14}}$ $\overline{\mathrm{s0d15}}$	in/out in/out in/out in/out in/out in/out in/out in/out	High byte of data bus in SCSI-W mode.
P20	s0oe	s0oe	out	External bus driver direction for $\overline{\mathrm{s0d0}}$ to $\overline{\mathrm{s0d7}}$ and $\overline{\mathrm{s0p}}$.
A20	s1oe	s1oe	out	External bus driver direction for $\overline{\mathrm{s0d8}}$ to $\overline{\mathrm{s0d15}}$ and $\overline{\mathrm{s1p}}$.
P19	$\overline{\mathrm{s0en}}$	$\overline{\mathrm{s0en}}$	out	External driver output enable for $\overline{\mathrm{s0ack}}$ and $\overline{\mathrm{s0atn}}$.
W17	pb4	$\overline{\mathrm{s0enph}}$	in/out	SCSI-0 phase enable for software ID select.
U16	pb7	s0enloid	in/out	Enable arbitration ID 15 - 8 for software ID.

Table 19-22 SCSI-W Signals

Note 6: If software ID is enabled, the host SCSI ID number is driven to external logic during arbitration. Please refer to the information about the software ID and external buffer solution in Chapter 10.

Note 7: These OR-gated SCSI bus signals are generated by the external SCSI buffer.

19.10.2 ATA

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\text{s0msg}}$	iordy	in	I/O ready.
T18	$\overline{\text{s0cd}}$	dmarq0	in	DMA request bus 0.
T19	$\overline{\text{s0io}}$	dmarq1	in	DMA request bus 1.
T20	$\overline{\text{s0req}}$	dmarq2	in	DMA request bus 2.
R18	$\overline{\text{s0rst}}$	dmarq3	in	DMA request bus 3.
P17	s0rst	$\overline{\text{cs0}}$	out	Chip select 0.
R19	$\overline{\text{s0ack}}$	$\overline{\text{cs1}}$	out	Chip select 1.
R20	$\overline{\text{s0atn}}$	a0	out	Device address bit 0.
K20	s0bsy	a1	out	Device address bit 1.
P20	s0oe	a2	out	Device address bit 2.
N18	$\overline{\text{s0p}}$	$\overline{\text{dmack0}}$	out	DMA acknowledge bus 0.
E19	$\overline{\text{s1p}}$	$\overline{\text{dmack1}}$	out	DMA acknowledge bus 1.
D20	$\overline{\text{s1msg}}$	$\overline{\text{dmack2}}$	out	DMA acknowledge bus 2.
E18	$\overline{\text{s1cd}}$	$\overline{\text{dmack3}}$	out	DMA acknowledge bus 3.
L20	$\overline{\text{s0d0}}$	d0	in/out	16-bit data bus of ATA port.
L18	$\overline{\text{s0d1}}$	d1	in/out	
L19	$\overline{\text{s0d2}}$	d2	in/out	
M19	$\overline{\text{s0d3}}$	d3	in/out	
M18	$\overline{\text{s0d4}}$	d4	in/out	
M17	$\overline{\text{s0d5}}$	d5	in/out	
N20	$\overline{\text{s0d6}}$	d6	in/out	
N19	$\overline{\text{s0d7}}$	d7	in/out	
G17	$\overline{\text{s1d0}}$	d8	in/out	
E20	$\overline{\text{s1d1}}$	d9	in/out	
F19	$\overline{\text{s1d2}}$	d10	in/out	
G18	$\overline{\text{s1d3}}$	d11	in/out	
G19	$\overline{\text{s1d4}}$	d12	in/out	
G20	$\overline{\text{s1d5}}$	d13	in/out	
H18	$\overline{\text{s1d6}}$	d14	in/out	
H19	$\overline{\text{s1d7}}$	d15	in/out	
P19	$\overline{\text{s0en}}$	$\overline{\text{dior0}}$	out	Read strobe signal 0.
K19	s0sel	$\overline{\text{dior1}}$	out	Read strobe signal 1.
B19	$\overline{\text{s1en}}$	$\overline{\text{dior2}}$	out	Read strobe signal 2.
H20	s1sel	$\overline{\text{dior3}}$	out	Read strobe signal 3.
K18	$\overline{\text{s0bsy}}$	intrq0	in	Interrupt request bus 0.
J20	$\overline{\text{s0sel}}$	intrq1	in	Interrupt request bus 1.
J19	$\overline{\text{s1bsy}}$	intrq2	in	Interrupt request bus 2.
J18	$\overline{\text{s1sel}}$	intrq3	in	Interrupt request bus 3.
C19	s1rst	$\overline{\text{diow0}}$	out	Write strobe signal 0.
B20	$\overline{\text{s1ack}}$	$\overline{\text{diow1}}$	out	Write strobe signal 1.
C18	$\overline{\text{s1atn}}$	$\overline{\text{diow2}}$	out	Write strobe signal 2.
J17	s1bsy	$\overline{\text{diow3}}$	out	Write strobe signal 3.
A20	s1oe	ext_oe	out	Output enable for external driver.

Table 19-23 ATA Signals

19.10.3 Parallel Ports

Parallel Port p0

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description (Note 8)
V20	$\overline{\text{s0msg}}$	p0perror	in	Peripheral error signal to parallel port p0.
T18	$\overline{\text{s0cd}}$	$\overline{\text{p0ack}}$	in	Peripheral acknowledgement signal to parallel port p0.
T19	$\overline{\text{s0io}}$	p0busy	in	Peripheral busy signal to parallel port p0.
T20	$\overline{\text{s0req}}$	$\overline{\text{p0fault}}$	in	Peripheral fault signal to parallel port p0.
R18	$\overline{\text{s0rst}}$	p0select	in	Peripheral select signal to parallel port p0.
P17	s0rst	p0data_oe	out	Data output enable from parallel port p0.
R19	$\overline{\text{s0ack}}$	$\overline{\text{p0selectin}}$	out	Select in signal from parallel port p0.
R20	$\overline{\text{s0atn}}$	$\overline{\text{p0autofd}}$	out	Autofeed signal from parallel port p0.
K20	s0bsy	$\overline{\text{p0strobe}}$	out	Strobe signal from parallel port p0.
P20	s0oe	$\overline{\text{p0init}}$	out	Initialization signal from parallel port p0.
L20 L18 L19 M19 M18 M17 N20 N19	$\overline{\text{s0d0}}$ $\overline{\text{s0d1}}$ $\overline{\text{s0d2}}$ $\overline{\text{s0d3}}$ $\overline{\text{s0d4}}$ $\overline{\text{s0d5}}$ $\overline{\text{s0d6}}$ $\overline{\text{s0d7}}$	p0d0 p0d1 p0d2 p0d3 p0d4 p0d5 p0d6 p0d7	in/out in/out in/out in/out in/out in/out in/out in/out	8-bit data bus of parallel port p0.

Table 19-24 Parallel Port p0 Signals

Parallel Port p1

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description (Note 8)
D20	$\overline{\text{s1msg}}$	p1perror	in	Peripheral error signal to parallel port p1.
E18	$\overline{\text{s1cd}}$	$\overline{\text{p1ack}}$	in	Peripheral acknowledgement signal to parallel port p1.
D19	$\overline{\text{s1io}}$	p1busy	in	Peripheral busy signal to parallel port p1.
E17	$\overline{\text{s1req}}$	$\overline{\text{p1fault}}$	in	Peripheral fault signal to parallel port p1.
C20	$\overline{\text{s1rst}}$	p1select	in	Peripheral select signal to parallel port p1.
C19	s1rst	p1data_oe	out	Data output enable from parallel port p1.
B20	$\overline{\text{s1ack}}$	$\overline{\text{p1selectin}}$	out	Select in signal from parallel port p1.
C18	$\overline{\text{s1atn}}$	$\overline{\text{p1autofd}}$	out	Autofeed signal from parallel port p1.
J17	s1bsy	$\overline{\text{p1strobe}}$	out	Strobe signal from parallel port p1.
A20	s1oe	$\overline{\text{p1init}}$	out	Initialization signal from parallel port p1.
G17 E20 F19 G18 G19 G20 H18 H19	$\overline{\text{s1d0}}$ $\overline{\text{s1d1}}$ $\overline{\text{s1d2}}$ $\overline{\text{s1d3}}$ $\overline{\text{s1d4}}$ $\overline{\text{s1d5}}$ $\overline{\text{s1d6}}$ $\overline{\text{s1d7}}$	p1d0 p1d1 p1d2 p1d3 p1d4 p1d5 p1d6 p1d7	in/out in/out in/out in/out in/out in/out in/out in/out	8-bit data bus of parallel port p1.

Table 19-25 Parallel Port p1 Signals

Note 8: These descriptions apply to Centronics (Compatibility) mode only, which is the default mode of the parallel ports. For descriptions of these signals in other IEEE 1284 modes, please refer to *chapter* 13 *Parallel Ports.*

Parallel Port-W (16-bit wide)

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\text{s0msg}}$	p0perror	in	Peripheral acknowledge signal to $\overline{\text{p0init}}$.
T18	$\overline{\text{s0cd}}$	$\overline{\text{p0ack}}$	in	Handshake signal in reverse mode.
T19	$\overline{\text{s0io}}$	p0busy	in	Flow control signal in the forward direction.
T20	$\overline{\text{s0req}}$	$\overline{\text{p0fault}}$	in	Interrupt request from peripheral.
R18	$\overline{\text{s0rst}}$	p0select	in	Indicates the ECP mode support.
P17	s0rst	p0data_oe	out	Data output enable signal.
R19	$\overline{\text{s0ack}}$	$\overline{\text{p0selectin}}$	out	Negotiation signal.
R20	$\overline{\text{s0atn}}$	$\overline{\text{p0autofd}}$	out	Flow control signal in the reverse direction.
K20	s0bsy	$\overline{\text{p0strobe}}$	out	Handshake signal in forward mode.
P20	s0oe	$\overline{\text{p0init}}$	out	Indicates the reverse mode when asserted (low).
L20	$\overline{\text{s0d0}}$	p0d0	in/out	16-bit data bus of parallel port-W.
L18	$\overline{\text{s0d1}}$	p0d1	in/out	
L19	$\overline{\text{s0d2}}$	p0d2	in/out	
M19	$\overline{\text{s0d3}}$	p0d3	in/out	
M18	$\overline{\text{s0d4}}$	p0d4	in/out	
M17	$\overline{\text{s0d5}}$	p0d5	in/out	
N20	$\overline{\text{s0d6}}$	p0d6	in/out	
N19	$\overline{\text{s0d7}}$	p0d7	in/out	
G17	$\overline{\text{s1d0}}$	p0d8	in/out	
E20	$\overline{\text{s1d1}}$	p0d9	in/out	
F19	$\overline{\text{s1d2}}$	p0d10	in/out	
G18	$\overline{\text{s1d3}}$	p0d11	in/out	
G19	$\overline{\text{s1d4}}$	p0d12	in/out	
G20	$\overline{\text{s1d5}}$	p0d13	in/out	
H18	$\overline{\text{s1d6}}$	p0d14	in/out	
H19	$\overline{\text{s1d7}}$	p0d15	in/out	

Table 19-26 Parallel Port-W Signals

19.10.4 Shared RAM and Shared RAM-W

Shared RAM

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\text{s0msg}}$	pr_adr0	in	Address bit 0, internally multiplexed with internally generated address bits.
T18	$\overline{\text{s0cd}}$	pr_adr1	in	Address bit 1, internally multiplexed with internally generated address bits.
T19	$\overline{\text{s0io}}$	$\overline{\text{intio}}$	in	Interrupt from peripheral.
T20	$\overline{\text{s0req}}$	rd_wr	in	Read/write select.
R18	$\overline{\text{s0rst}}$	$\overline{\text{pr\_req}}$	in	Request from peripheral.
P17	s0rst	$\overline{\text{pr\_int}}$	out	Interrupt to peripheral.
R19	$\overline{\text{s0ack}}$	$\overline{\text{pr\_ack}}$	out	Acknowledgement to peripheral.
R20	$\overline{\text{s0atn}}$	a_sel	out	Address select for externally multiplexed address bits. High for address from external device.
L20 L18 L19 M19 M18 M17 N20 N19	$\overline{\text{s0d0}}$ $\overline{\text{s0d1}}$ $\overline{\text{s0d2}}$ $\overline{\text{s0d3}}$ $\overline{\text{s0d4}}$ $\overline{\text{s0d5}}$ $\overline{\text{s0d6}}$ $\overline{\text{s0d7}}$	pr_d0 pr_d1 pr_d2 pr_d3 pr_d4 pr_d5 pr_d6 pr_d7	in/out in/out in/out in/out in/out in/out in/out in/out	Data bus.

Table 19-27 Shared RAM Signals

Shared RAM-W

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V20	$\overline{\text{s0msg}}$	pr_adr0	in	Address bit 0, internally multiplexed with internally generated address bits.
T18	$\overline{\text{s0cd}}$	pr_adr1	in	Address bit 1, internally multiplexed with internally generated address bits.
T19	$\overline{\text{s0io}}$	$\overline{\text{intio}}$	in	Interrupt from peripheral.
T20	$\overline{\text{s0req}}$	rd_wr	in	Read/write select.
R18	$\overline{\text{s0rst}}$	$\overline{\text{pr\_req}}$	in	Request from peripheral.
P17	s0rst	$\overline{\text{pr\_int}}$	out	Interrupt to peripheral.
R19	$\overline{\text{s0ack}}$	$\overline{\text{pr\_ack}}$	out	Acknowledgement to peripheral.
R20	$\overline{\text{s0atn}}$	a_sel	out	Address select for externally multiplexed address bits. High for address from external device.
L20	$\overline{\text{s0d0}}$	pr_d0	in/out	Data bus low byte.
L18	$\overline{\text{s0d1}}$	pr_d1	in/out	
L19	$\overline{\text{s0d2}}$	pr_d2	in/out	
M19	$\overline{\text{s0d3}}$	pr_d3	in/out	
M18	$\overline{\text{s0d4}}$	pr_d4	in/out	
M17	$\overline{\text{s0d5}}$	pr_d5	in/out	
N20	$\overline{\text{s0d6}}$	pr_d6	in/out	
N19	$\overline{\text{s0d7}}$	pr_d7	in/out	
G17	$\overline{\text{s1d0}}$	pr_d8	in/out	Data bus high byte.
E20	$\overline{\text{s1d1}}$	pr_d9	in/out	
F19	$\overline{\text{s1d2}}$	pr_d10	in/out	
G18	$\overline{\text{s1d3}}$	pr_d11	in/out	
G19	$\overline{\text{s1d4}}$	pr_d12	in/out	
G20	$\overline{\text{s1d5}}$	pr_d13	in/out	
H18	$\overline{\text{s1d6}}$	pr_d14	in/out	
H19	$\overline{\text{s1d7}}$	pr_d15	in/out	

Table 19-28 Shared RAM-W Signals

19.10.5 Asynchronous Serial Ports

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
U19	txd1	txd1	out	Transmit data from Asynchronous Serial Port p1.
W20	$\overline{\text{rts1}}$	$\overline{\text{rts1}}$	out	Request to send from Asynchronous Serial Port p1.
T17	rxd1	rxd1	in	Receive data at Asynchronous Serial Port p1.
V19	$\overline{\text{cts1}}$	$\overline{\text{cts1}}$	in	Clear to send at Asynchronous Serial Port p1.
P19	$\overline{\text{s0en}}$	$\overline{\text{rts2}}$	out	Request to send from Asynchronous Serial Port p2.
K19	s0sel	txd2	out	Transmit data to Asynchronous Serial Port p2.
K18	$\overline{\text{s0bsy}}$	$\overline{\text{cts2}}$	in	Clear to send to Asynchronous Serial Port p2.
J20	$\overline{\text{s0sel}}$	rxd2	in	Receive data at Asynchronous Serial Port p2.
J19	$\overline{\text{s1bsy}}$	$\overline{\text{cts3}}$	in	Clear to send to Asynchronous Serial Port p3.
J18	$\overline{\text{s1sel}}$	rxd3	in	Receive data at Asynchronous Serial Port p3.
B19	$\overline{\text{s1en}}$	$\overline{\text{rts3}}$	out	Request to send from Asynchronous Serial Port p3.
H20	s1sel	txd3	out	Transmit data from Asynchronous Serial Port p3.

Table 19-29 Asynchronous Serial Ports p1, p2 and p3 Signals

19.10.6 Synchronous Serial Ports p1 and p3

The synchronous serial ports have six different modes of operation:

Master Output;
Master Input;

Slave Output;
Slave Input;

Master Bidirectional;
Slave Bidirectional.

The signal names at the I/O pins differ, depending upon the mode in use. The following tables show the different I/O pin assignments of the two synchronous serial ports in each mode of operation.

SYNCHRONOUS SERIAL PORTS - MASTER OUTPUT MODE							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	-	J18	$\overline{\text{s1sel}}$	-	ss1_in1 ss3_in1	Not used by the synchronous serial ports in Master Output mode.
V19	$\overline{\text{cts1}}$	ss1status	J19	$\overline{\text{s1bsy}}$	ss3status	ss1_in2 ss3_in2	Serial busy input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1data	B19	$\overline{\text{s1en}}$	ss3data	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1frame	U16	pb7	ss3frame	ss1_io3 ss3_io3	Serial frame indicator output from respective port.

Table 19-30 Pin Assignments of Synchronous Serial Ports p1 and p3 in Master Output Mode

SYNCHRONOUS SERIAL PORTS - MASTER INPUT MODE							
Synchronous Serial Port p1			**Synchronous Serial Port p3**				
Solder Ball	**Chip Pin Name**	**Mode Signal Name**	**Solder Ball**	**Chip Pin Name**	**Mode Signal Name**	**Interface Signal Name**	**Description**
T17	rxd1	ss1data	J18	$\overline{\text{s1sel}}$	ss3data	ss1_in1 ss3_in1	Serial data input to respective port.
V19	$\overline{\text{cts1}}$	ss1status	J19	$\overline{\text{s1bsy}}$	ss3status	ss1_in2 ss3_in2	Serial empty input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1frame	B19	$\overline{\text{s1en}}$	ss3frame	ss1_out2 ss3_out2	Serial frame indicator from respective port.
W17	pb4	-	U16	pb7	-	-	Not used by the synchronous serial ports in Master Input mode.

Table 19-31 Pin Assignments of Synchronous Serial Ports p1 and p3 in Master Input Mode

SYNCHRONOUS SERIAL PORTS - SLAVE OUTPUT MODE							
Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	Chip Pin Name	Mode Signal Name	Solder Ball	Chip Pin Name	Mode Signal Name	Interface Signal Name	Description
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	ss1data	H20	s1sel	ss3data	ss1_out1 ss3_out1	Serial data output from respective port.
W20	$\overline{\text{rts1}}$	ss1status	B19	$\overline{\text{s1en}}$	ss3status	ss1_out2 ss3_out2	Serial empty output from respective port.
W17	pb4	-	U16	pb7	-	-	Not used by the synchronous serial ports in Slave Output mode.

Table 19-32 Pin Assignments of Synchronous Serial Ports p1 and p3 in Slave Output Mode

SYNCHRONOUS SERIAL PORTS - SLAVE INPUT MODE							
Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	Chip Pin Name	Mode Signal Name	Solder Ball	Chip Pin Name	Mode Signal Name	Interface Signal Name	Description
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	-	H20	s1sel	-	ss1_out1 ss3_out1	Not used by the synchronous serial ports in Slave Input mode.
W20	$\overline{\text{rts1}}$	ss1status	B19	$\overline{\text{s1en}}$	ss3status	ss1_out2 ss3_out2	Serial busy output from respective port.
W17	pb4	ss1data	U16	pb7	ss3data	ss1_io3 ss3_io3	Serial data input to respective port.

Table 19-33 Pin Assignments of Synchronous Serial Ports p1 and p3 in Slave Input Mode

SYNCHRONOUS SERIAL PORTS - MASTER BIDIRECTIONAL MODE							
Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	Chip Pin Name	Mode Signal Name	Solder Ball	Chip Pin Name	Mode Signal Name	Interface Signal Name	Description
T17	rxd1	ss1status	J18	$\overline{\text{s1sel}}$	ss3status	ss1_in1 ss3_in1	Serial busy input to respective port.
V19	$\overline{\text{cts1}}$	ss1idata	J19	$\overline{\text{s1bsy}}$	ss3idata	ss1_in2 ss3_in2	Serial data input to respective port.
U19	txd1	ss1clk	H20	s1sel	ss3clk	ss1_out1 ss3_out1	Serial clock output from respective port.
W20	$\overline{\text{rts1}}$	ss1odata	B19	$\overline{\text{s1en}}$	ss3odata	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1frame	U16	pb7	ss3frame	ss1_io3 ss3_io3	Serial frame indicator output from respective port.

Table 19-34 Pin Assignments of Synchronous Serial Ports p1 and p3 in Master Bidirectional Mode

SYNCHRONOUS SERIAL PORTS - SLAVE BIDIRECTIONAL MODE							
Synchronous Serial Port p1			Synchronous Serial Port p3				
Solder Ball	Chip Pin Name	Mode Signal Name	Solder Ball	Chip Pin Name	Mode Signal Name	Interface Signal Name	Description
T17	rxd1	ss1clk	J18	$\overline{\text{s1sel}}$	ss3clk	ss1_in1 ss3_in1	Serial clock input to respective port.
V19	$\overline{\text{cts1}}$	ss1frame	J19	$\overline{\text{s1bsy}}$	ss3frame	ss1_in2 ss3_in2	Serial frame indicator to respective port.
U19	txd1	ss1status	H20	s1sel	ss3status	ss1_out1 ss3_out1	Serial busy output from respective port.
W20	$\overline{\text{rts1}}$	ss1odata	B19	$\overline{\text{s1en}}$	ss3odata	ss1_out2 ss3_out2	Serial data output from respective port.
W17	pb4	ss1idata	U16	pb7	ss3idata	ss1_io3 ss3_io3	Serial data input to respective port.

Table 19-35 Pin Assignments of Synchronous Serial Ports p1 and p3 in Slave Bidirectional Mode

19.10.7 USB

USB Port p1

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
U19	txd1	$\overline{\text{usb1\_oe}}$	out	Output enable from USB Port p1.
W20	$\overline{\text{rts1}}$	usb1_speed	out	Speed indicator signal from USB Port p1.
T17	rxd1	usb1_rcv	in	Serial data input to USB Port p1.
V19	$\overline{\text{cts1}}$	usb1_vp	in	Plus (D+) input to USB Port p1.
Y17	pb2	usb1_vpo	out	Plus (D+) output from USB Port p1.
V16	pb3	usb1_vmo	out	Minus (D-) output from USB Port p1.
Y18	pb5	usb1_vm	in	Minus (D-) input to USB Port p1.

Table 19-36 USB Port p1 Signals

USB Port p2

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
D20	$\overline{\text{s1msg}}$	$\overline{\text{usb2\_oe}}$	out	Output enable from USB Port p2.
E18	$\overline{\text{s1cd}}$	usb2_speed	out	Speed indicator signal from USB Port p2.
D19	$\overline{\text{s1io}}$	usb2_rcv	in	Serial data input to USB Port p2.
E17	$\overline{\text{s1req}}$	usb2_vp	in	Plus (D+) input to USB Port p2.
C18	$\overline{\text{s1atn}}$	usb2_vpo	out	Plus (D+) output from USB Port p2.
C20	$\overline{\text{s1rst}}$	usb2_vm	in	Minus (D-) input to USB Port p2.
J17	s1bsy	usb2_vmo	out	Minus (D-) output from USB Port p2.

Table 19-37 USB Port p2 Signals

19.10.8 Chip Selects for Peripherals (CSP)

The CSP port offers additional chip select signals for use with peripheral devices.

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
Y17	pb2	$\overline{\text{csp1}}$	out	Additional peripheral chip select signal 1.
V16	pb3	$\overline{\text{csp2}}$	out	Additional peripheral chip select signal 2.
W17	pb4	$\overline{\text{csp3}}$	out	Additional peripheral chip select signal 3.
Y18	pb5	$\overline{\text{csp5}}$	out	Additional peripheral chip select signal 5.
Y19	pb6	$\overline{\text{csp6}}$	out	Additional peripheral chip select signal 6.
U16	pb7	$\overline{\text{csp7}}$	out	Additional peripheral chip select signal 7.

Table 19-38 CSP Signals

19.10.9 I2C

Solder Ball	Chip Pin Name	Interface Pin Name	Direction	Description
V15	pb0	i2c_d	in/out	I2C data input/output.
W16	pb1	i2c_clk	out	I2C output clock signal.

Table 19-39 I2C Signals

19.10.10 General Port PB

Solder Ball	Name	Direction	Description
V15	pb0	in/out	General Port PB, bit 0.
W16	pb1	in/out	General Port PB, bit 1.
Y17	pb2	in/out	General Port PB, bit 2.
V16	pb3	in/out	General Port PB, bit 3.
W17	pb4	in/out	General Port PB, bit 4.
Y18	pb5	in/out	General Port PB, bit 5.
Y19	pb6	in/out	General Port PB, bit 6.
U16	pb7	in/out	General Port PB, bit 7.

Table 19-40 General Port PB Signals

19.11 I/O Pin Default Values

All bidirectional ports are input by default. The output pins have the default values given in the table below.

Solder Ball	Default Value	Chip Pin Name	Interface Pin Name						
			SCSI	Serial Ports	ATA	Parallel Ports	Shared RAM	Gen I/O	USB
P17	0	s0rst	s0rst		cs0	p0data_oe	pr_int	g5	
R19	0	s0ack	s0ack		cs1	p0selectin	pr_ack	g4	
R20	0	s0atn	s0atn		a0	p0autofd	a_sel	g3	
K20	0	s0bsy	s0bsy		a1	p0strobe		g2	
P20	0	s0oe	s0oe		a2	p0init		g1	
P19	1	s0en	s0en	rts2	dior0			g7	
K19	No default (Note 9)	s0sel	s0sel	txd2	dior1			g6	
B19	1	s1en	s1en	rts3	dior2			g31	
H20	No default (Note 9)	s1sel	s1sel	txd3	dior3			g30	
C19	0	s1rst	s1rst		diow0	p1data_oe		g29	
B20	0	s1ack	s1ack		diow1	p1selectin		g28	
C18	0	s1atn	s1atn		diow2	p1autofd		g27	
J17	0	s1bsy	s1bsy		diow3	p1strobe		g26	
A20	0	s1oe	s1oe		ext_oe	p1init		g25	
Y7	1	txd0		txd0					
V8	1	rts0		rts0					
U19	1	txd1		txd1					usb_oe
W20	1	rts1		rts1					usb1_speed

Table 19-41 I/O Pin Default Values

Note 9: These pins have no default. Their condition depends upon the state of serial port hardware configuration signal hcfg as follows:

if (hcfg = 1), then {s0sel = 0, s1sel = 0}
if (hcfg = 0), then {s0sel = 1, s1sel = 1}

Signal hcfg is listed in Table 19-7.

19.12 DC Electrical Specifications

19.12.1 Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
V_{DD}	DC supply voltage.	0	3.6	V
V_{in}	DC input voltage.	0	6.5	V
V_{out}	DC off-state output voltage.	0	6.5	V
T_{stg}	Storage temperature.	-65	150	ºC
T_A	Operating temperature.	0	70	ºC

Table 19-42 Absolute Maximum Ratings

ESD protection: up to 2kV (Human Body Model according to the MIL-STD-883 method 3015).

Lead temperature: accords with the specification for a JEDEC Level 3 package.

19.12.2 Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
V_{DD}	DC supply voltage.	3.0	3.3	3.6	V
V_{in}	DC input voltage.	0	-	5.5	V
V_{out}	DC off-state output voltage.	0	-	5.5	V
T_A	Operating temp. (Ambient temp. range).	0	-	70	ºC
I_{OH}	High level output current: pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7. All other.	 - -	-	 -12 -4	 mA mA
I_{OL}	Low level output current: pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7. All other.	 - -	-	 12 4	 mA mA

Table 19-43 Recommended Operating Conditions

19.12.3 Capacitance

All pins have a typical capacitance of 5 pF and a maximum capacitance of 6 pF.

19.12.4 DC Electrical Characteristics

Symbol	Parameter	Min	Typical	Max	Unit
V_{IH}	High level input voltage, all inputs except clkin.	2.0	-	5.5	V
V_{IL}	Low level input voltage, all inputs except clkin.	0	-	0.8	V
V_{IH}	High level input voltage, clkin.	0.7 x V_{DD}	-	5.5	V
V_{IL}	Low level input voltage, clkin.	0	-	0.3 x V_{DD}	V
	Schmitt-trigger hysteresis. (See Table 19-45).	0.5	0.575	0.65	V
V_{OH}	High level output voltage.	2.4	-	V_{DD}	V
V_{OL}	Low level output voltage.	0	-	0.4	V
I_{in}	Input leakage current.	-10	-	10	µA
I_{ioz}	I/O leakage current.	-10	-	10	µA
I_{DD}	Supply current.	-	105	170	mA

Table 19-44 DC Electrical Characteristics

19.12.5 Input Buffer Types

Signal	Buffer Type
clkin.	CMOS
$\overline{cts0}$, $\overline{irq}$, $\overline{nmi}$, $\overline{reset}$, rxd0.	TTL Schmitt Trigger
All other.	TTL

Table 19-45 Input Buffer Types

19.13 AC Electrical Specifications

This section provides the AC characteristics for the ETRAX 100LX. The timing sequences are related to the internal 100 MHz clock.

The table below lists all bus states used in the timing diagrams on the following pages.

Bus State Descriptions

State	Description	Comment
T_a	Activate state. The $\overline{rd}$, $\overline{wr}$ or $\overline{inta}$ strobes are asserted in this state. Signal $\overline{cas}$ is asserted after the end of this state.	
T_d	Data state. “Data in” is sampled at the end of this state, except for EDO DRAM, where data is sampled 15 ns after the end of this state.	
T_z	Data bus turn-off state. This state is inserted between bursts, to allow the ETRAX 100LX and external units to turn off their outputs before the data bus is driven by another source. This state may overlap with a T_{ew} state.	
T_{pa}	Row address signal $\overline{ras}$ precharge activate state. Signal $\overline{ras}$ is set high after the end of this state.	
T_{pd}	Row address signal $\overline{ras}$ precharge state. DRAM row address is asserted after the end of this state. During DRAM refresh, column address signal $\overline{cas}$ is set low after the end of this state.	
T_{ra}	Row address signal $\overline{ras}$ activate state. Signal $\overline{ras}$ is set low after the end of this state.	
T_{rd}	Row address hold state.	
T_{be0},T_{be1}	Burst end states. This is inserted at the end of an EDO DRAM read burst, to allow the last data of the burst to be sampled. These states may overlap with a T_{ew} state.	
T_{ew}	Early wait state. This state may overlap with a T_z , T_{zw} , T_{be0}, or T_{be1} state.	Inserted before T_a
T_{lw}	Late wait state.	Inserted between T_a and T_d
T_{zw}	Turn-off wait state. This state may overlap with a T_{ew} state.	Inserted after T_d
T_{xw}	External wait state.	
T_{xx}	Any bus state.	

Table 19-46 Bus State Descriptions

Wait States

The table below lists the wait state parameters used in the timing diagrams that follow. These parameters correspond to the wait state values described in Chapter 5.

Name	Description
ew	Number of early wait states.
lw	Number of late wait states.
zw	Number of turn-off wait states.
c	Column address signal $\overline{cas}$ delay.
cw	Number of $\overline{cas}$ wait states.
cp	Number of $\overline{cas}$ precharge wait states.
rp	Number of $\overline{ras}$ precharge wait states.
rs	Number of row address setup wait states.
rh	Number of row address hold wait states.
cz	Number of turn-off wait states after $\overline{cas}$.

Table 19-47 Wait States

19.13.1 Conditions

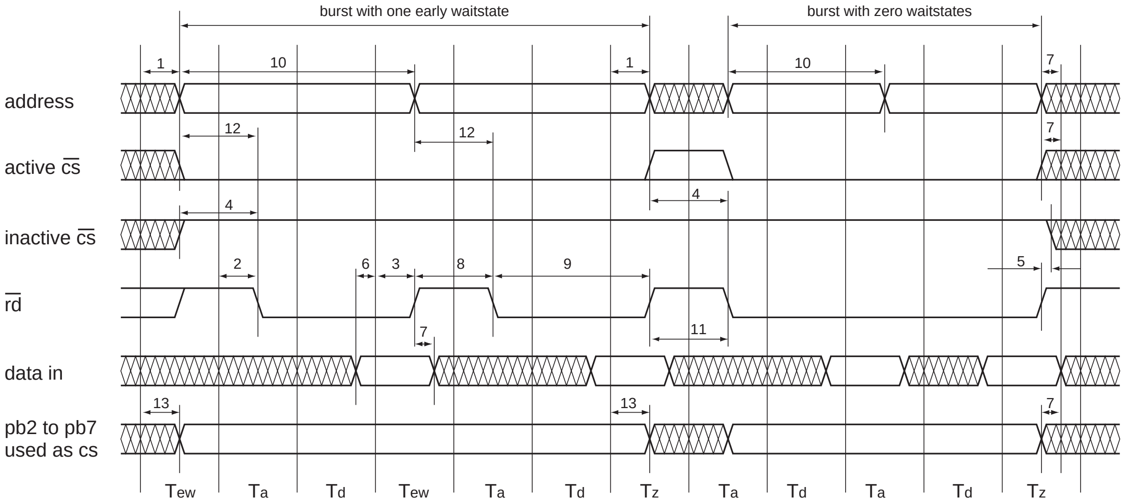
The timing information in the chapter is valid under the operating conditions given in the table below.

Condition	Value
T_A	0°C to 70°C
V_{DD}	3.3 V +/- 0.3 V
Capacitive load	50 pF

Table 19-48 Operating Conditions for Timing Information

19.13.2 SRAM/Flash/Peripheral Timing

Read Cycle



Active and inactive $\overline{cs}$ are valid for $\overline{cse0}$, $\overline{cse1}$, $\overline{csr0}$, $\overline{csr1}$, $\overline{csp0}$ and $\overline{csp4}$.

Figure 19-2 SRAM/Flash/Peripheral - Read Cycle Timing Diagram

No	Name	Explanation	Without Waitstates			Add with Waitstates	Unit
			Min	Nom	Max		
1	t_a	Address and chip select delay from clock. (Note 10).	2	-	8	-	ns
2	t_{rl}	Read low delay from clock.	2	-	8	-	ns
3	t_{rh}	Read high delay from clock.	2	-	7	-	ns
4	t_{cshr}	Chip select high to read low.	8	-	-	-	ns
5	t_{rhcs}	Read high to chip select low.	0	-	-	-	ns
6	t_{ds}	Data in setup time to clock.	0	-	-	-	ns
7	t_{dh}	Data hold time from address, chip select or read, whichever occurs first.	0	-	-	-	ns
8	t_{rhw}	Read inactive width within burst.	-2	-	-	10·ew	ns
9	t_{rw}	Read active width.	16	20	-	10·lw	ns
10	t_{rc}	Read cycle.	-	20	-	10·(ew+lw)	ns
11	t_{rhr}	Read inactive width after burst.	8	-	-	10·zw	ns
12	t_{ar}	Read inactive time after chip select or address.	-2	-	-	10·ew	ns
13	t_{pcs}	pb delay from clock when used as chip selects.	3	-	12	-	ns

Table 19-49 SRAM/Flash/Peripheral Read Cycle Timing

Note 10: Valid for $\overline{cse0}$, $\overline{cse1}$, $\overline{csr0}$, $\overline{csr1}$, $\overline{csp0}$ and $\overline{csp4}$.

Write Cycle, Normal Write

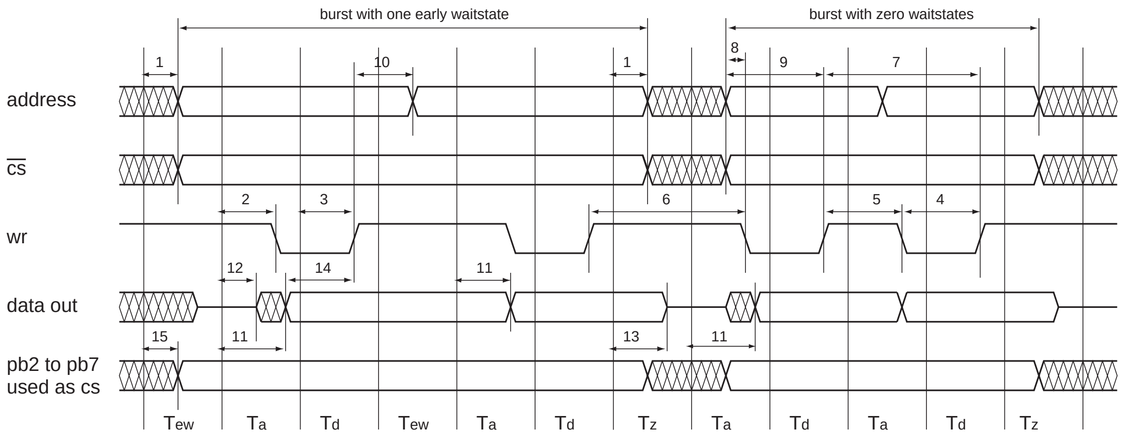


Figure 19-3 SRAM/Flash/Peripheral Write Cycle Timing Diagram - Normal Write

No	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_a	Address and chip select delay from clock.	2	-	8	-	ns
2	t_{wl}	Write low delay from clock.	7	-	13	-	ns
3	t_{wh}	Write high delay from clock.	7	-	13	-	ns
4	t_{ww}	Write pulse width.	6	-	-	10·lw	ns
5	t_{whw}	Write inactive width within burst.	9	-	-	10·ew	ns
6	t_{whl}	Write inactive width after burst.	19	-	-	10·zw	ns
7	t_{wc}	Write cycle time.	-	20	-	10·(ew+lw)	ns
8	t_{awl}	Address and chip select setup to write low.	2	-	-	10·ew	ns
9	t_{awh}	Address and chip select setup to end of write.	11	-	-	10·(ew+lw)	ns
10	t_{ahw}	Address hold after write high.	3	-	-	-	ns
11	t_{do}	Data delay from clock.	6	-	13	-	ns
12	t_{doe}	Data turn on time from clock.	6	-	-	-	ns
13	t_{doz}	Data turn off time from clock.	6	-	10	10·zw	ns
14	t_{dwh}	Data valid to end of write.	6	-	-	10·lw	ns
15	t_{pcs}	pb delay from clock, when used as chip selects.	3	-	12	-	ns

Table 19-50 SRAM/Flash/Peripheral Write Cycle Timing - Normal Write

Write Cycle, Extended Write

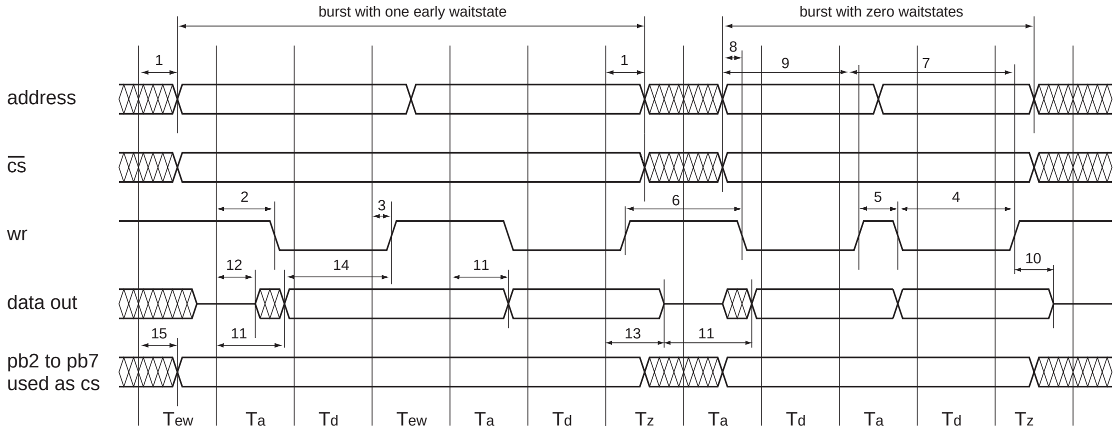


Figure 19-4 SRAM/Flash/Peripheral Write Cycle Timing Diagram - Extended Write

No	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_a	Address and chip select delay from clock.	2	-	8	-	ns
2	t_{wl}	Write low delay from clock.	7	-	13	-	ns
3	t_{whx}	Write high delay from clock.	2	-	7	-	ns
4	t_{wwx}	Write pulse width.	11	-	-	10·lw	ns
5	t_{whwx}	Write inactive width within burst.	4	-	-	10·ew	ns
6	t_{whlx}	Write inactive width after burst.	13	-	-	10·zw	ns
7	t_{wcx}	Write cycle time.	-	20	-	10·(ew+lw)	ns
8	t_{awl}	Address and chip select setup to write low.	2	-	-	10·ew	ns
9	t_{awhx}	Address and chip select setup to end of write.	16	-	-	10·(ew+lw)	ns
10	t_{whdx}	Data hold after end of write.	2	-	-	10·ew within burst, 10·zw after burst	ns
11	t_{do}	Data delay from clock.	6	-	13	-	ns
12	t_{doe}	Data turn on time from clock.	6	-	-	-	ns
13	t_{doz}	Data turn off time from clock.	6	-	10	10·zw	ns
14	t_{dwhx}	Data valid to end of write.	11	-	-	10·lw	ns
15	t_{pcs}	pb delay from clock, when used as chip selects.	3	-	12	-	ns

Table 19-51 SRAM/Flash/Peripheral Write Cycle Timing - Extended Write Timing

19.13.3 Synchronous DRAM

50/100 MHz Mode, Read

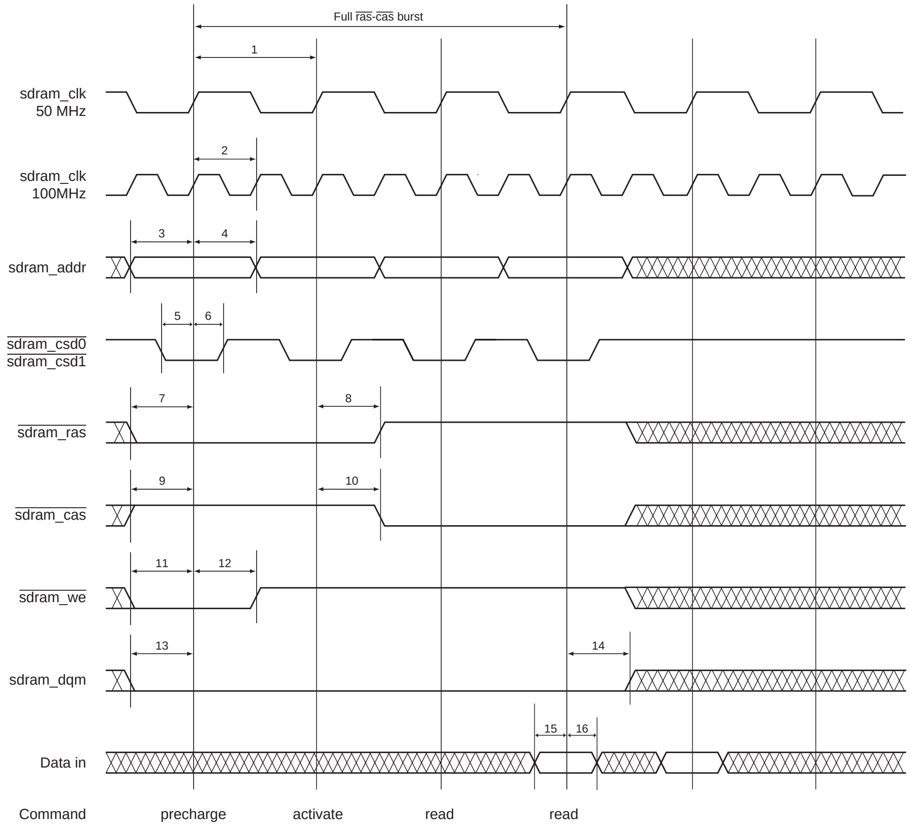


Figure 19-5 Synchronous DRAM, 50/100 MHz Mode - Read Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{sclks}	Clock period, 50 MHz mode	-	20	-	ns
2	t_{sclkf}	Clock period, 100 MHz mode	-	10	-	ns
3	t_{sas}	Address setup time to clock	9	-	-	ns
4	t_{sah}	Address hold time from clock	9	-	-	ns
5	t_{scs}	sdram_csd0/1 setup time to clock	4	-	-	ns
6	t_{ch}	sdram_csd0/1 hold time from clock.	4	-	-	ns
7	t_{rs}	sdram_ras setup time to clock	9	-	-	ns
8	t_{rh}	sdram_ras hold time from clock	9	-	-	ns
9	t_{scs}	sdram_cas setup time to clock	9	-	-	ns
10	t_{sch}	sdram_cas hold time from clock	9	-	-	ns
11	t_{sws}	sdram_we setup time to clock	9	-	-	ns
12	t_{swh}	sdram_we hold time from clock	9	-	-	ns
13	t_{sds}	sdram_dqm setup time to clock	9	-	-	ns
14	t_{sdh}	sdram_dqm hold time from clock	9	-	-	ns
15	t_{sdis}	Data in setup time to clock	1	-	-	ns
16	t_{sdih}	Data in hold time from clock	3	-	-	ns

Table 19-52 Synchronous DRAM, 50/100 MHz Mode - Read Timing

50/100 MHz Mode, Write

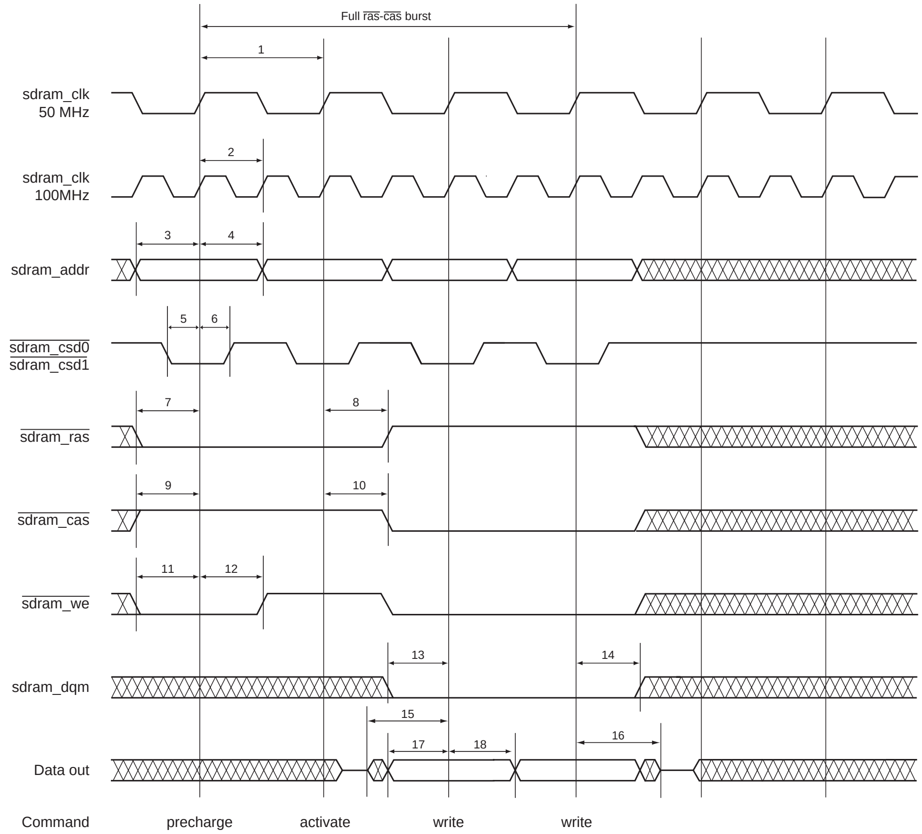


Figure 19-6 Synchronous DRAM, 50/100 MHz Mode - Write Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{sclks}	Clock period, 50 MHz mode	-	20	-	ns
2	t_{sclkf}	Clock period, 100 MHz mode	-	10	-	ns
3	t_{sas}	Address setup time to clock	9	-	-	ns
4	t_{sah}	Address hold time from clock	9	-	-	ns
5	t_{scs}	$\overline{\text{sdram\_csd0/1}}$ setup time to clock	4	-	-	ns
6	t_{sch}	$\overline{\text{sdram\_csd0/1}}$ hold time from clock.	4	-	-	ns
7	t_{srs}	$\overline{\text{sdram\_ras}}$ setup time to clock	9	-	-	ns
8	t_{srh}	$\overline{\text{sdram\_ras}}$ hold time from clock	9	-	-	ns
9	t_{scs}	$\overline{\text{sdram\_cas}}$ setup time to clock	9	-	-	ns
10	t_{sch}	$\overline{\text{sdram\_cas}}$ hold time from clock	9	-	-	ns
11	t_{sws}	$\overline{\text{sdram\_we}}$ setup time to clock	9	-	-	ns
12	t_{swh}	$\overline{\text{sdram\_we}}$ hold time from clock	9	-	-	ns
13	t_{sds}	sdram_dqm setup time to clock	9	-	-	ns
14	t_{sdh}	sdram_dqm hold time from clock	9	-	-	ns
15	t_{sdoe}	Data out turn on time to clock	-	-	11	ns
16	t_{sdoz}	Data out turn off time from clock	-	-	13	ns
17	t_{sdos}	Data out setup time to clock	7	-	-	ns
18	t_{sdoh}	Data out hold time from clock	9	-	-	ns

Table 19-53 Synchronous DRAM, 50/100 MHz Mode - Write Timing

DDR 100 MHz Mode, Read

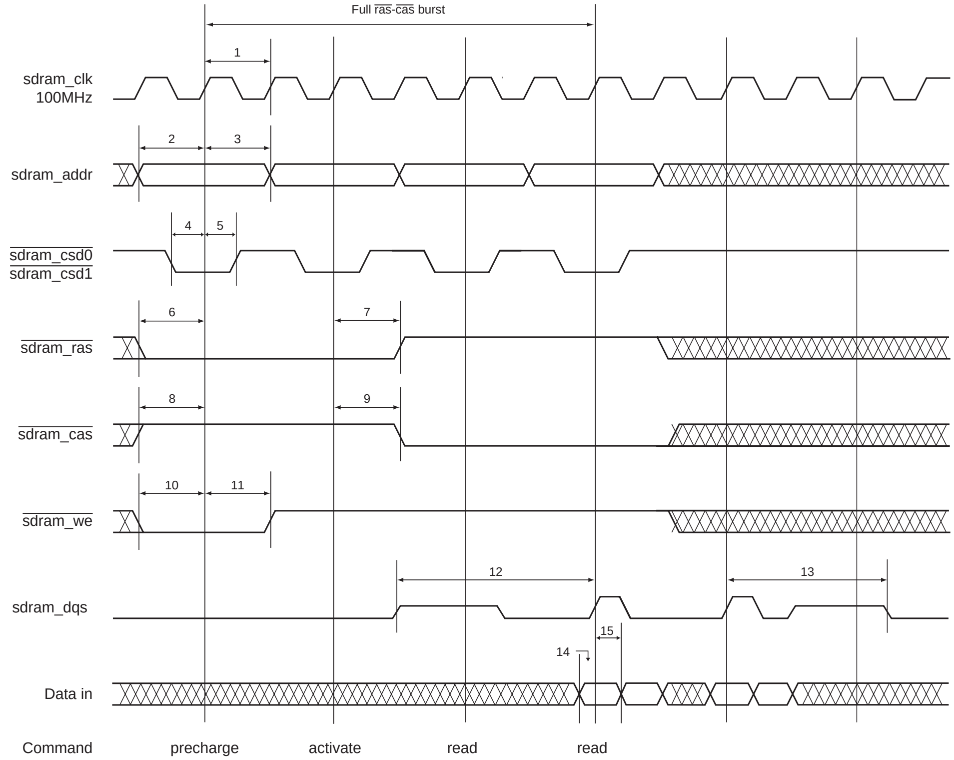


Figure 19-7 DDR 100 MHz Mode - Read Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{sclkf}	Clock period	-	10	-	ns
2	t_{sas}	Address setup time to clock	9	-	-	ns
3	t_{sah}	Address hold time from clock	9	-	-	ns
4	t_{scs}	$\overline{\text{sdram\_csd0/1}}$ setup time to clock	4	-	-	ns
5	t_{sch}	$\overline{\text{sdram\_csd0/1}}$ hold time from clock.	4	-	-	ns
6	t_{srs}	$\overline{\text{sdram\_ras}}$ setup time to clock	9	-	-	ns
7	t_{srh}	$\overline{\text{sdram\_ras}}$ hold time from clock	9	-	-	ns
8	t_{scs}	$\overline{\text{sdram\_cas}}$ setup time to clock	9	-	-	ns
9	t_{sch}	$\overline{\text{sdram\_cas}}$ hold time from clock	9	-	-	ns
10	t_{sws}	$\overline{\text{sdram\_we}}$ setup time to clock	9	-	-	ns
11	t_{swh}	$\overline{\text{sdram\_we}}$ hold time from clock	9	-	-	ns
12	t_{dsdq}	sdram_dqs turn off time to clock	22	-	-	ns
13	t_{dsdqe}	sdram_dqs turn on time from clock	24	-	-	ns
14	t_{dsdis}	Data in setup time from sdram_dqs	-0.5	-	-	ns
15	t_{dsdih}	Data in hold time from sdram_dqs	3	-	-	ns

Table 19-54 DDR 100 MHz Mode - Read Timing

DDR 100 MHz Mode, Write

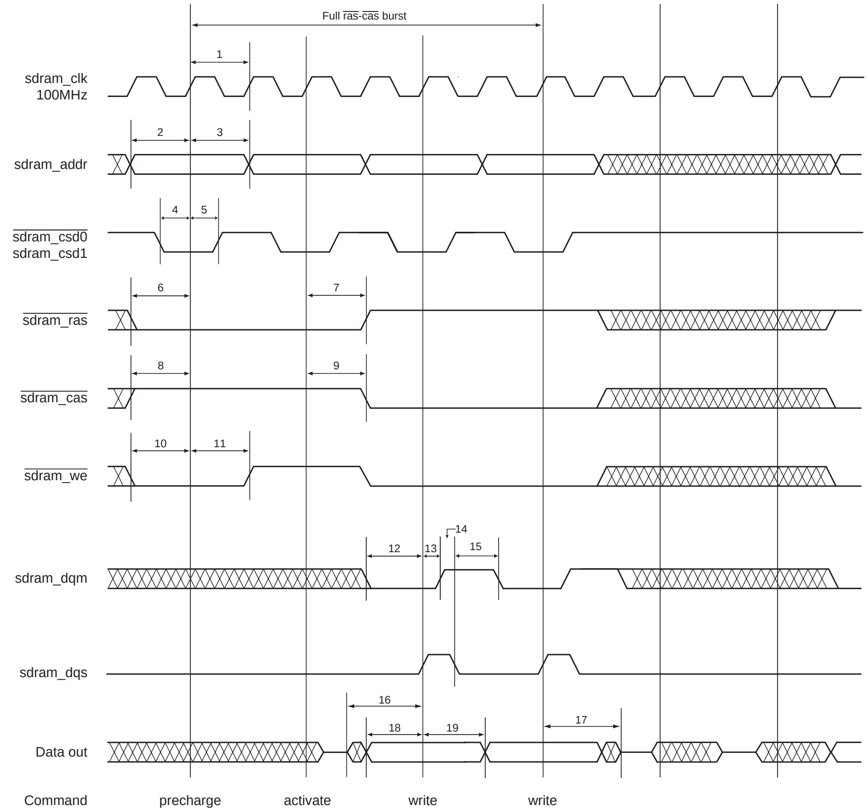


Figure 19-8 DDR 100 MHz Mode - Write Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{sclkf}	Clock period	-	10	-	ns
2	t_{sas}	Address setup time to clock	9	-	-	ns
3	t_{sah}	Address hold time from clock	9	-	-	ns
4	t_{scs}	$\overline{\text{sdram\_csd0/1}}$ setup time to clock	4	-	-	ns
5	t_{sch}	$\overline{\text{sdram\_csd0/1}}$ hold time from clock.	4	-	-	ns
6	t_{srs}	$\overline{\text{sdram\_ras}}$ setup time to clock	9	-	-	ns
7	t_{srh}	$\overline{\text{sdram\_ras}}$ hold time from clock	9	-	-	ns
8	t_{scs}	$\overline{\text{sdram\_cas}}$ setup time to clock	9	-	-	ns
9	t_{sch}	$\overline{\text{sdram\_cas}}$ hold time from clock	9	-	-	ns
10	t_{sws}	$\overline{\text{sdram\_we}}$ setup time to clock	9	-	-	ns
11	ts_{wh}	$\overline{\text{sdram\_we}}$ hold time from clock	9	-	-	ns
12	t_{dsdmsr}	sdram_dqm setup time to sdram_dqs rising edge	6	-	-	ns
13	t_{dsdmhr}	sdram_dqm hold time from sdram_dqs rising edge	2.5	-	-	ns
14	t_{dsdmsf}	sdram_dqm setup time to sdram_dqs falling edge	1.5	-	-	ns
15	t_{dsdmhf}	sdram_dqm hold time to sdram_dqs falling edge	8	-	-	ns
16	t_{dsdoe}	Data out turn on time to sdram_dqs	-	-	28	ns
17	t_{dsdoz}	Data out turn off time from sdram_dqs	-	-	3	ns
18	t_{dsdos}	Data out setup time to sdram_dqs	8	-	-	ns
19	t_{dsdoh}	Data out hold time from sdram_dqs	3	-	-	ns

Table 19-55 DDR 100 MHz Mode - Write Timing

19.13.4 Asynchronous DRAM

Fast Page Mode, Read

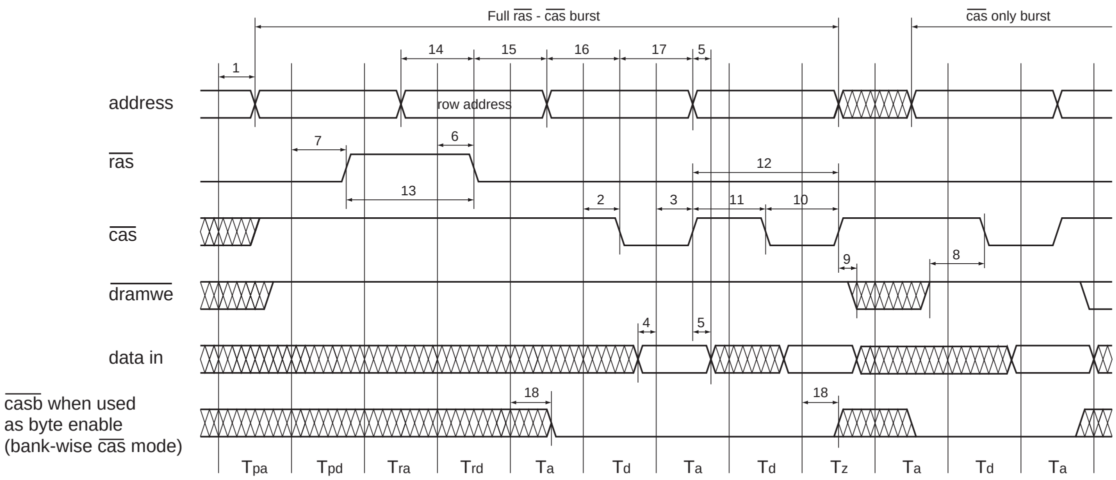


Figure 19-9 Asynchronous DRAM, Fast Page Mode - Read Timing Diagram

No	Name	Explanation	Without Waitstates Min	Without Waitstates Nom	Without Waitstates Max	Add with Waitstates	Unit
1	t_{a}	Address delay from clock.	2	-	8	-	ns
2	t_{casl}	cas low delay from clock.	2	-	8	5·c	ns
3	t_{cash}	cas high delay from clock.	2	-	8	-	ns
4	t_{ds}	Data in setup time to clock.	0	-	-	-	ns
5	t_{dh}	Data in hold time from cas or address change, whichever occurs first.	0	-	-	-	ns
6	t_{rasl}	ras low time from clock.	3	-	10	-	ns
7	t_{rash}	ras high time from clock.	7	-	13	-	ns
8	t_{rcs}	dramwe high setup time to cas low.	4	-	-	5·c	ns
9	t_{rch}	dramwe high hold time from cas high.	2	-	-	-	ns
10	t_{cas}	cas pulse width.	7	-	-	10·cw - 5·c	ns
11	t_{cp}	cas precharge time.	7	-	-	10·cp + 5·c	ns
12	t_{pc}	cas cycle time.	-	20	-	10·(cp + cw)	ns
13	t_{rp}	ras precharge time.	14	-	-	10·(rp + rs)	ns
14	t_{asr}	Row address setup time to ras.	9	-	-	10·rs	ns
15	t_{rah}	Row address hold time from ras.	4	-	-	10·rh	ns
16	t_{asc}	Column address setup time to cas.	7	-	-	10·cp + 5·c	ns
17	t_{cah}	Column address hold time from cas.	7	-	-	10·cw - 5·c	ns
18	t_{be}	casb delay from clock, when used as byte enable (bankwise mode).	2	-	8	-	ns

Table 19-56 Asynchronous DRAM, Fast Page Mode - Read Timing

EDO DRAM, Read

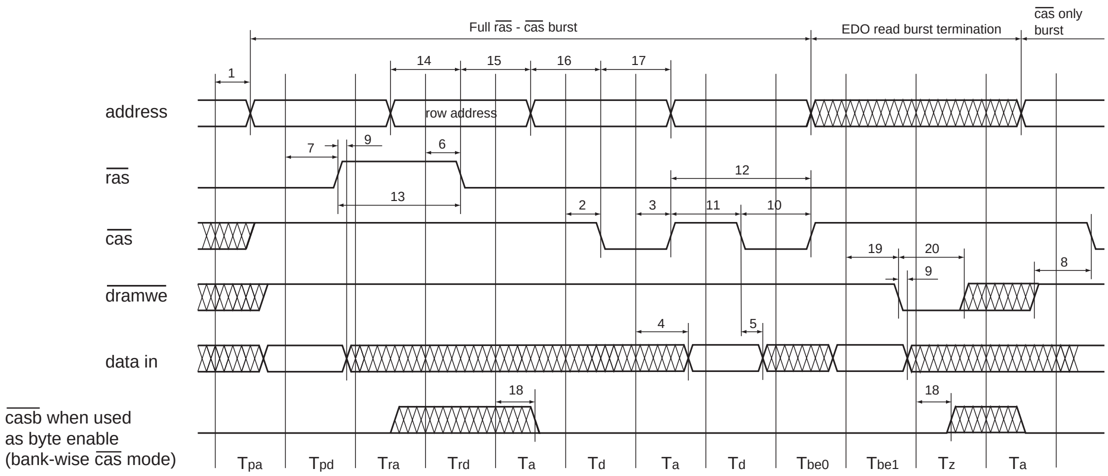


Figure 19-10 EDO DRAM - Read Timing Diagram

No	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_{a}	Address delay from clock.	2	-	8	-	ns
2	t_{casl}	$\overline{cas}$ low delay from clock.	2	-	8	5·c	ns
3	t_{cash}	$\overline{cas}$ high delay from clock.	2	-	8	-	ns
4	t_{dd}	Data in valid delay from clock.	-	-	15	-	ns
5	t_{coh}	Data in hold time from $\overline{cas}$ low.	4	-	-	(-10)·cp - 5·c	ns
6	t_{rasl}	$\overline{ras}$ low time from clock.	3	-	10	-	ns
7	t_{rash}	$\overline{ras}$ high time from clock.	8	-	13	-	ns
8	t_{rcs}	$\overline{dramwe}$ high setup time to $\overline{cas}$ low.	4	-	-	5·c	ns
9	t_{dh}	Data in hold time from $\overline{dramwe}$ low or $\overline{ras}$ high, whichever occurs first.	0	-	-	-	ns
10	t_{cas}	$\overline{cas}$ pulse width.	7	-	-	10·cw - 5·c	ns
11	t_{cp}	$\overline{cas}$ precharge time.	7	-	-	10·cp + 5·c	ns
12	t_{pc}	$\overline{cas}$ cycle time.	-	20	-	10·(cp + cw)	ns
13	t_{rp}	$\overline{ras}$ precharge time.	14	-	-	10·(rp + rs)	ns
14	t_{asr}	Row address setup time to $\overline{ras}$.	9	-	-	10·rs	ns
15	t_{rah}	Row address hold time from $\overline{ras}$.	4	-	12	10·rh	ns
16	t_{asc}	Column address setup time to $\overline{cas}$.	7	-	-	10·cp + 5·c	ns
17	t_{cah}	Column address hold time from $\overline{cas}$.	7	-	-	10·cw - 5·c	ns
18	t_{be}	$\overline{casb}$ delay from clock, when used as byte enable (bankwise mode).	2	-	8	-	ns
19	t_{wel}	$\overline{dramwe}$ low delay from clock.	7	-	13	-	ns
20	t_{wew}	$\overline{dramwe}$ pulse width after EDO read burst.	7	-	-	10·cz	ns

Table 19-57 EDO DRAM - Read Timing

Asynchronous DRAM, Write

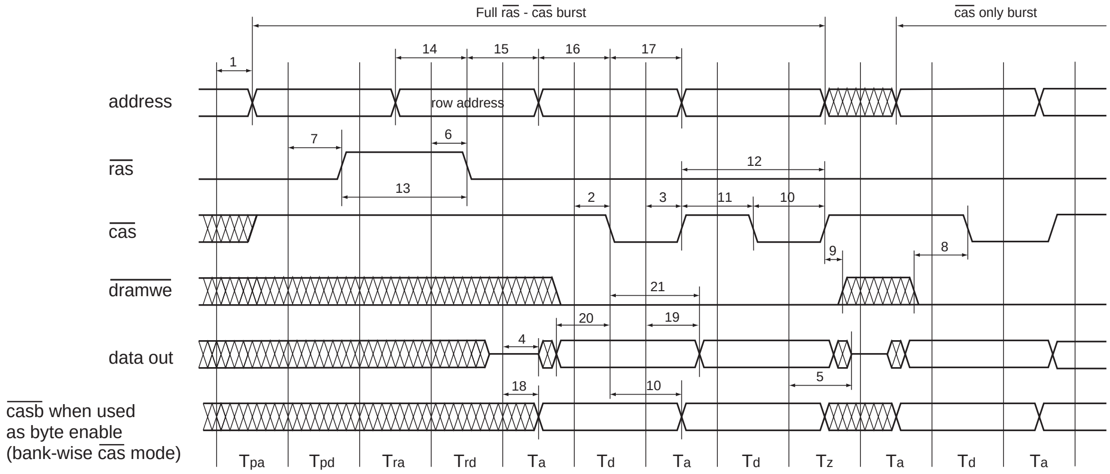


Figure 19-11 Asynchronous DRAM - Write Timing Diagram

No	Name	Explanation	Without Waitstates			Add with Waitstates	Unit
			Min	Nom	Max		
1	t_a	Address delay from clock.	2	-	8	-	ns
2	t_{casl}	$\overline{cas}$ low delay from clock.	2	-	8	5·c	ns
3	t_{cash}	$\overline{cas}$ high delay from clock.	2	-	8	-	ns
4	t_{doe}	Data out turn on time from clock.	7	-	-	-	ns
5	t_{doz}	Data out turn off time from clock.	-	-	10	10·cz	ns
6	t_{rasl}	$\overline{ras}$ low time from clock.	3	-	10	-	ns
7	t_{rash}	$\overline{ras}$ high time from clock.	8	-	13	-	ns
8	t_{wcs}	$\overline{dramwe}$ low setup time to $\overline{cas}$ low.	2	-	-	-	ns
9	t_{weh}	$\overline{dramwe}$ low hold time from $\overline{cas}$ high.	1	-	-	-	ns
10	t_{cas}	$\overline{cas}$ pulse width.	7	-	-	10·cw - 5·c	ns
11	t_{cp}	$\overline{cas}$ precharge time.	7	-	-	10·cp + 5·c	ns
12	t_{pc}	$\overline{cas}$ cycle time.	-	20	-	10·(cp + cw)	ns
13	t_{rp}	$\overline{ras}$ precharge time.	14	-	-	10·(rp + rs)	ns
14	t_{asr}	Row address setup time to $\overline{ras}$.	9	-	-	10·rs	ns
15	t_{rah}	Row address hold time from $\overline{ras}$.	4	-	12	10·rh	ns
16	t_{asc}	Column address setup time to $\overline{cas}$.	7	-	-	10·cp + 5·c	ns
17	t_{cah}	Column address hold time from $\overline{cas}$.	7	-	-	10·cw - 5·c	ns
18	t_{be}	$\overline{casb}$ delay from clock, when used as byte enable (bankwise mode).	2	-	8	-	ns
19	t_{do}	Data delay from clock.	6	-	-	-	ns
20	t_{dsc}	Data setup to $\overline{cas}$ low.	2	-	-	5·c	ns
21	t_{dhc}	Data hold after $\overline{cas}$ low.	10	-	-	10·cw - 5·c	ns

Table 19-58 Asynchronous DRAM - Write Timing

$\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh Cycle

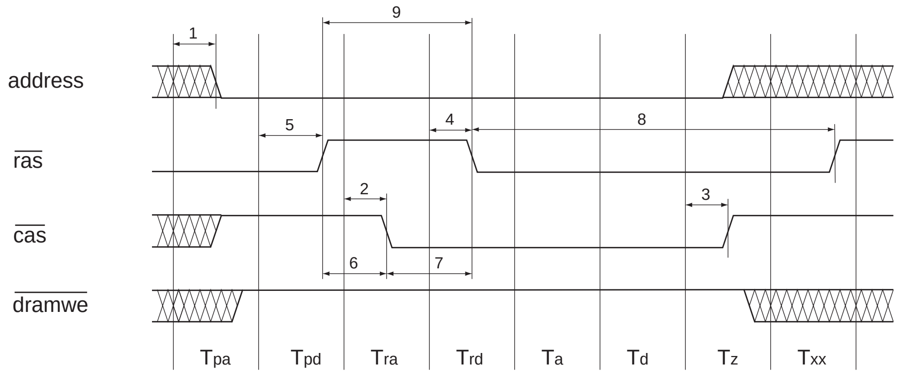


Figure 19-12 $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh Cycle Timing Diagram

No	Name	Explanation	Without Waitstates			Add with Waitstates	Unit
			Min	Nom	Max		
1	t_a	Address delay from clock.	2	-	8	-	ns
2	t_{casl}	$\overline{\text{cas}}$ low delay from clock.	2	-	8	5·c	ns
3	t_{cash}	$\overline{\text{cas}}$ high delay from clock.	2	-	8	-	ns
4	t_{rasl}	$\overline{\text{ras}}$ low time from clock.	3	-	10	-	ns
5	t_{rash}	$\overline{\text{ras}}$ high time from clock.	8	-	13	-	ns
6	t_{rpc}	$\overline{\text{ras}}$ to $\overline{\text{cas}}$ precharge time.	2	-	-	10·rp	ns
7	t_{csr}	$\overline{\text{cas}}$ setup time to $\overline{\text{ras}}$.	7	-	-	10·rs	ns
8	t_{ras}	$\overline{\text{ras}}$ pulse width.	40	-	-	10·(rh + cp + cw)	ns
9	t_{rp}	$\overline{\text{ras}}$ precharge time.	14	-	-	10·(rp + rs)	ns

Table 19-59 $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh Cycle Timing

19.13.5 General Bus Interface Timing Diagrams

Data Turn-off Timing

This diagram specifies the relationships of the data turn-off timing of Asynchronous DRAM and non-DRAM bursts.

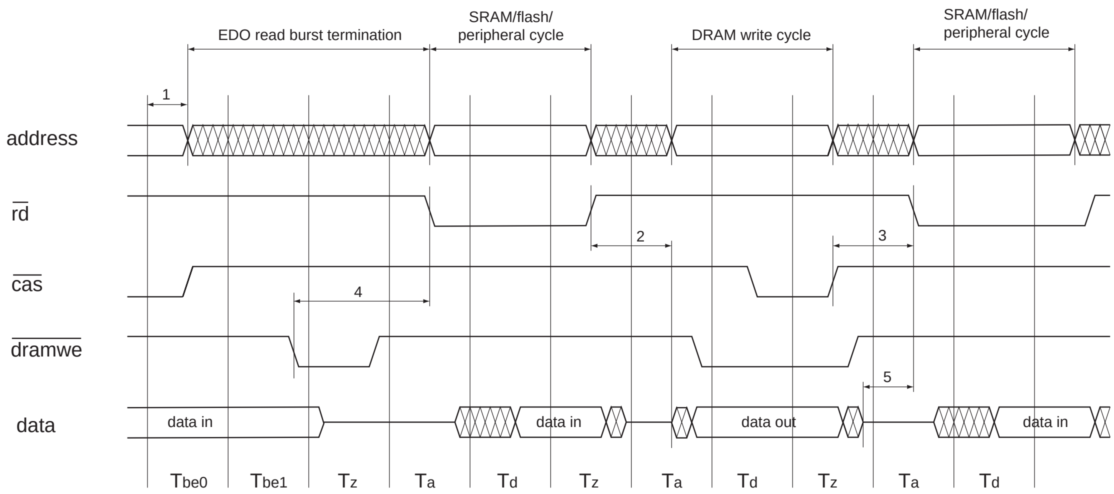


Figure 19-13 Data Turn-off Timing Diagram

No	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_a	Address delay from clock.	2	-	8	-	ns
2	t_{rdo}	Read high to data out.	13	-	-	10·zw	ns
3	t_{cr}	$\overline{\text{cas}}$ high to read low.	7	-	-	10·cz	ns
4	t_{welr}	$\overline{\text{dramwe}}$ low to read low after EDO read burst.	13	-	-	10·cz	ns
5	t_{dozr}	Data out turn-off before read low.	5	-	-	-	ns

Table 19-60 Data Turn-off Timing

External Interrupt Acknowledge Cycle

The external interrupt acknowledge cycle always uses a maximum number of waitstates.

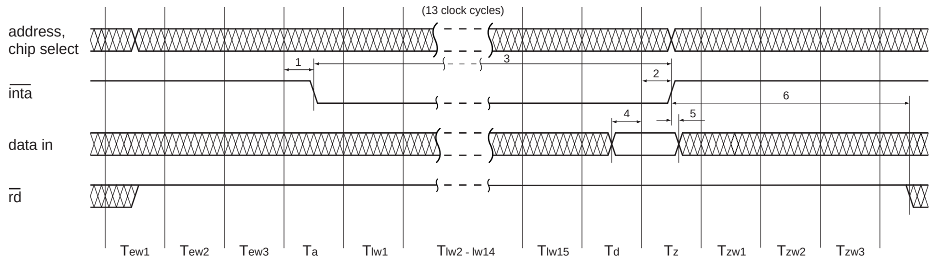


Figure 19-14 External Interrupt Acknowledge Cycle Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{il}	$\overline{\text{inta}}$ low delay from clock.	2	-	8	ns
2	t_{ih}	$\overline{\text{inta}}$ high delay from clock.	2	-	7	ns
3	t_{iw}	$\overline{\text{inta}}$ pulse width.	167	170	172	ns
4	t_{ds}	Data in setup time to clock.	0	-	-	ns
5	t_{dh}	Data in hold time from $\overline{\text{inta}}$ high.	0	-	-	ns
6	t_{ihr}	$\overline{\text{inta}}$ high to read low.	38	-	-	ns

Table 19-61 External Interrupt Acknowledge Cycle Timing

Timing of $\overline{\text{wait}}$ and $\overline{\text{rerun}}$

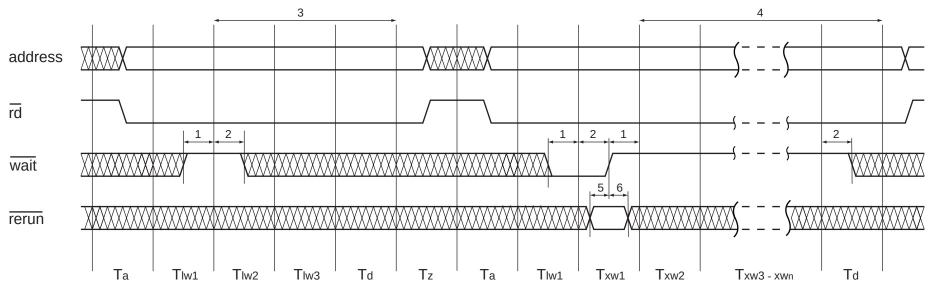


Figure 19-15 Wait and Rerun Timing Diagram

No	Name	Explanation	Min	Nom	Max	Note	Unit
1	t_{xws}	$\overline{\text{wait}}$ setup time to clock.	1	-	-	Note 11	ns
2	t_{xwh}	$\overline{\text{wait}}$ hold time from clock.	2	-	-	Note 11	ns
3	t_{xwi}	$\overline{\text{wait}}$ sampled to end of bus cycle ($\overline{\text{wait}}$ not activated).	30	30	30	Note 12	ns
4	t_{xwa}	$\overline{\text{wait}}$ sampled to end of bus cycle ($\overline{\text{wait}}$ activated).	50	-	80	Note 12	ns
5	t_{res}	$\overline{\text{rerun}}$ setup time to $\overline{\text{wait}}$ high.	1	-	-	-	ns
6	t_{reh}	$\overline{\text{rerun}}$ hold time after $\overline{\text{wait}}$ high.	1	-	-	-	ns

Table 19-62 Wait and Rerun Timing

Note 11: The $\overline{\text{wait}}$ signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

Note 12: The $\overline{\text{wait}}$ signal is sampled three clock cycles before the end of the bus cycle, taking only the internal wait states into consideration. To recognize $\overline{\text{wait}}$, a number of internal lw and/or ew waitstates must be added. Typically, three internal wait states are necessary, but this depends on the external wait state logic.

19.13.6 External DMA Timing Diagrams

External DMA, Handshake Mode Timing, Read

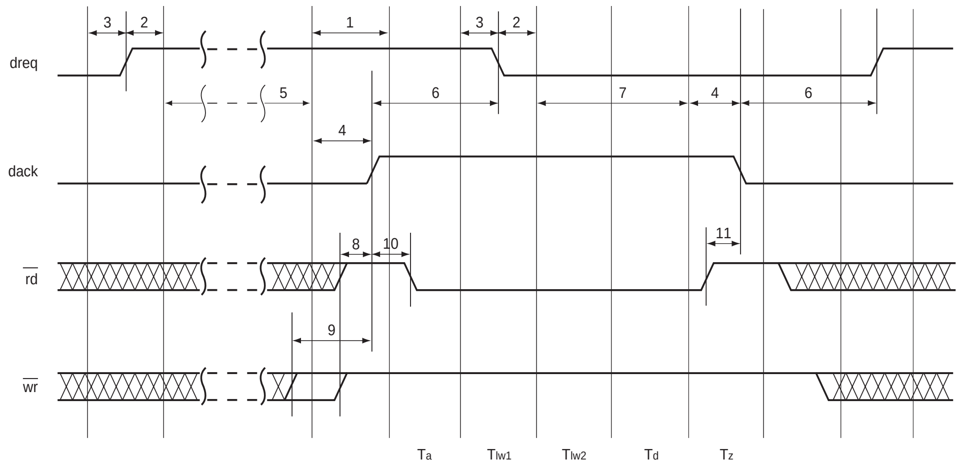


Figure 19-16 External DMA, Handshake Mode Timing, Read

The timing diagram above is shown with dreq and dack configured to be active high, and with two late waitstates (The number of waitstates is just an example however; there does not need to be two waitstates.).

Number	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_{ckp}	Internal clock period	-	10	-	-	ns
2	t_{drs}	dreq setup time to clock (note 13)	1	-	-	-	ns
3	t_{drh}	dreq hold time from clock (note 13)	2	-	-	-	ns
4	t_{dack}	dack delay from clock	8	-	17	-	ns
5	t_{drda}	dreq to dack latency	4*t_{ckp}	-	-	-	ns
6	t_{dadr}	dreq hold time from dack	0	-	-	-	ns
7	t_{dai}	dreq inactive to dack inactive latency (note 14)	2*t_{ckp}	-	-	-	ns
8	t_{rwda}	$\overline{wr}$ (extended write) high or $\overline{rd}$ high to dack active	5	-	-	-	ns
9	t_{wrda}	$\overline{wr}$ (normal write) high to dack active	10	-	-	-	ns
10	t_{dard}	dack active to $\overline{rd}$ low	1	-	-	10*max(0,ew-1)	ns
11	t_{rhda}	$\overline{rd}$ high to dack inactive	5	-	-	-	ns

Table 19-63 External DMA, Handshake Mode Timing, Read

Note 13: The dreq signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

Note 14: t_{dai} will be the maximum of 2*t_{ckp} and the time to the end of the Td state of the external DMA bus cycle.

External DMA, Handshake Mode Timing, Write

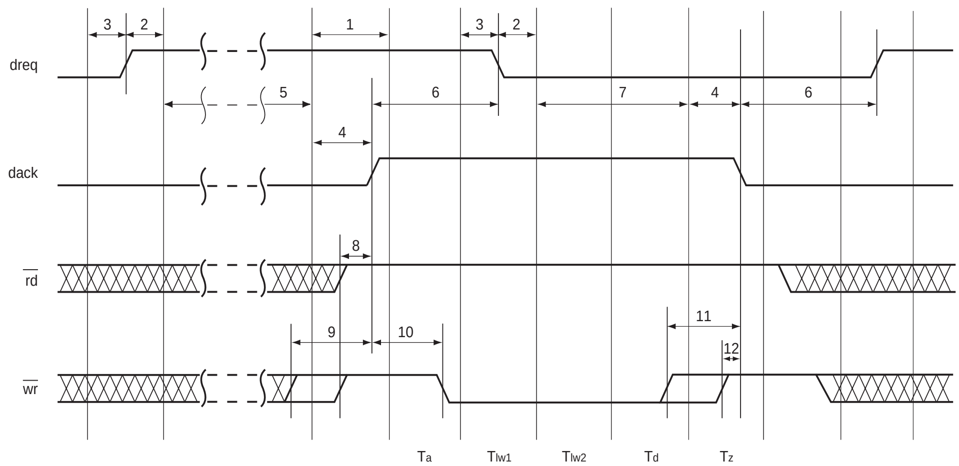


Figure 19-17 External DMA, Handshake Mode Timing, Write

The timing diagram above is shown with dreq and dack configured to be active high, and with two late waitstates (The number of waitstates is just an example however; there does not need to be two waitstates.).

Number	Name	Explanation	Without Waitstates			Add with Waitstates	Unit
			Min	Nom	Max		
1	t_{ckp}	Internal clock period	-	10	-	-	ns
2	t_{drs}	dreq setup time to clock (note 15)	1	-	-	-	ns
3	t_{drh}	dreq hold time from clock (note 15)	2	-	-	-	ns
4	t_{dack}	dack delay from clock	8	-	17	-	ns
5	t_{drda}	dreq to dack latency	4*t_{ckp}	-	-	-	ns
6	t_{dadr}	dreq hold time from dack	0	-	-	-	ns
7	t_{dai}	dreq inactive to dack inactive latency (note 16)	2*t_{ckp}	-	-	-	ns
8	t_{rwda}	$\overline{wr}$ (extended write) high or $\overline{rd}$ high to dack active	5	-	-	-	ns
9	t_{wrda}	$\overline{wr}$ (normal write) high to dack active	10	-	-	-	ns
10	t_{dawr}	dack active to $\overline{wr}$ low	6	-	-	10*max(0,ew-1)	ns
11	t_{whda}	$\overline{wr}$ (normal write) high to dack inactive	10	-	-	-	ns
12	t_{weda}	$\overline{wr}$ (extended write) high to dack inactive	5	-	-	-	ns

Table 19-64 External DMA, Handshake Mode Timing, Write

Note 15: The dreq signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

Note 16: t_{dai} will be the maximum of 2*t_{ckp} and the time to the end of the Td state of the external DMA bus cycle.

External DMA, Burst Mode Timing, Read

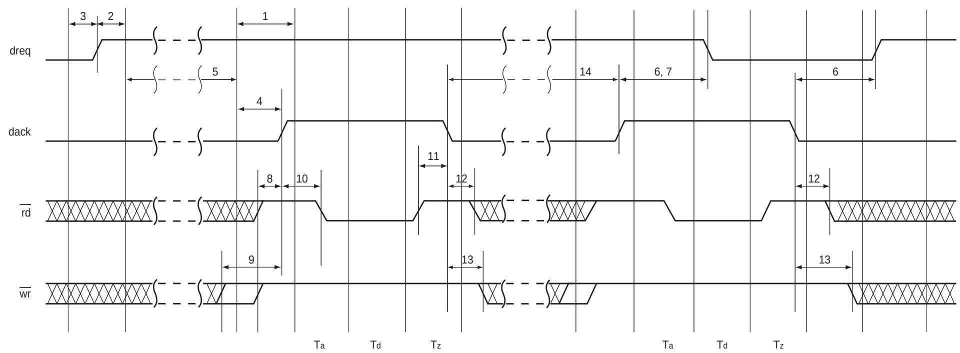


Figure 19-18 External DMA, Burst Mode Timing, Read

The timing diagram above is shown with dreq and dack configured to be active high, and with 0 waitstates.

Number	Name	Explanation	Without Waitstates Min	Nom	Max	Add with Waitstates	Unit
1	t_{ckp}	Internal clock period	-	10	-	-	ns
2	t_{drs}	dreq setup time to clock (note 17)	1	-	-	-	ns
3	t_{drh}	dreq hold time from clock (note 17)	2	-	-	-	ns
4	t_{dack}	dack delay from clock	8	-	17	-	ns
5	t_{drda}	dreq to dack latency	$4*t_{ckp}$	-	-	-	ns
6	t_{dadr}	dreq hold time from dack	0	-	-	-	ns
7	t_{dadi}	dreq inactive time from dack	-	-	12	10*(max(0,ew-1)+lw)	ns
8	t_{rwda}	$\overline{wr}$ (extended write) high or $\overline{rd}$ high to dack active	5	-	-	-	ns
9	t_{wrda}	$\overline{wr}$ (normal write) high to dack active	10	-	-	-	ns
10	t_{dard}	dack active to $\overline{rd}$ low	1	-	-	10*max(0,ew-1)	ns
11	t_{rhda}	$\overline{rd}$ high to dack inactive	5	-	-	-	ns
12	t_{dair}	dack inactive to $\overline{rd}$ low	1	-	-	10*zw	ns
13	t_{daiw}	dack inactive to $\overline{wr}$ low	6	-	-	10*zw	ns
14	t_{dada}	dack inactive to dack active time	17	-	-	-	ns

Table 19-65 External DMA, Burst Mode Timing, Read

Note 17: The dreq signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

External DMA, Burst Mode Timing, Write

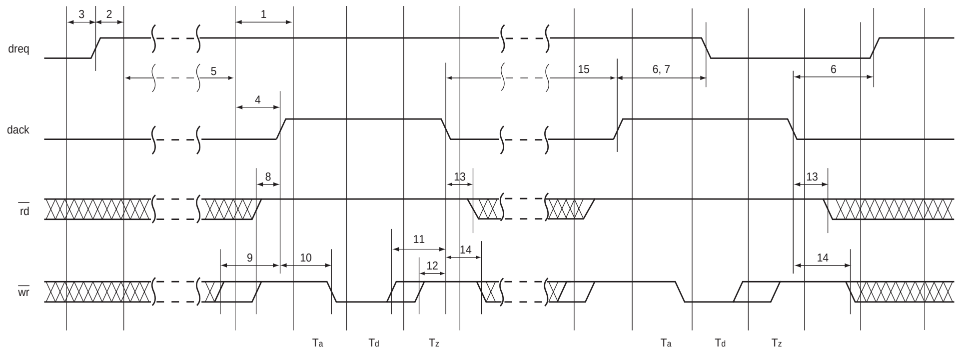


Figure 19-19 External DMA, Burst Mode Timing, Write

The timing diagram above is shown with dreq and dack configured to be active high, and with 0 waitstates.

Number	Name	Explanation	Without Waitstates			Add with Waitstates	Unit
			Min	Nom	Max		
1	t_{ckp}	Internal clock period	-	10	-	-	ns
2	t_{drs}	dreq setup time to clock (note 18)	1	-	-	-	ns
3	t_{drh}	dreq hold time from clock (note 18)	2	-	-	-	ns
4	t_{dack}	dack delay from clock	8	-	17	-	ns
5	t_{drda}	dreq to dack latency	4*t_{ckp}	-	-	-	ns
6	t_{dadr}	dreq hold time from dack	0	-	-	-	ns
7	t_{dadi}	dreq inactive time from dack	-	-	12	10*(max(0,ew-1)+lw)	ns
8	t_{rwda}	$\overline{wr}$ (extended write) high or $\overline{rd}$ high to dack active	5	-	-	-	ns
9	t_{wrda}	$\overline{wr}$ (normal write) high to dack active	10	-	-	-	ns
10	t_{dawr}	dack active to $\overline{wr}$ low	6	-	-	10*max(0,ew-1)	ns
11	t_{whda}	$\overline{wr}$ (normal write) high to dack inactive	10	-	-	-	ns
12	t_{weda}	$\overline{wr}$ (extended write) high to dack inactive	5	-	-	-	ns
13	t_{dair}	dack inactive to $\overline{rd}$ low	1	-	-	10*zw	ns
14	t_{daiw}	dack inactive to $\overline{wr}$ low	6	-	-	10*zw	ns
15	t_{dada}	dack inactive to dack active time	17	-	-	-	ns

Table 19-66 External DMA, Burst Mode Timing, Write

Note 18: The dreq signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

19.13.7 $\overline{irq}$ and $\overline{nmi}$ Timing

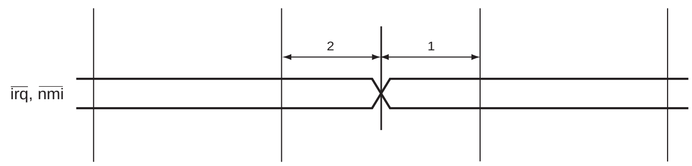


Figure 19-20 $\overline{irq}$ and $\overline{nmi}$ Timing

Number	Name	Explanation	Min	Nom	Max	Unit
1	t_{irqs}	$\overline{irq}$ and $\overline{nmi}$ setup time to clock (note 19)	6	-	-	ns
2	t_{irqh}	$\overline{irq}$ and $\overline{nmi}$ hold time from clock (note 19)	-3	-	-	ns

Table 19-67 $\overline{irq}$ and $\overline{nmi}$ Timing

Note 19: The $\overline{irq}$ and $\overline{nmi}$ signals are synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

19.13.8 Shared RAM Interface Timing

Shared RAM Interface, Peripheral Read and Write Cycle Timing

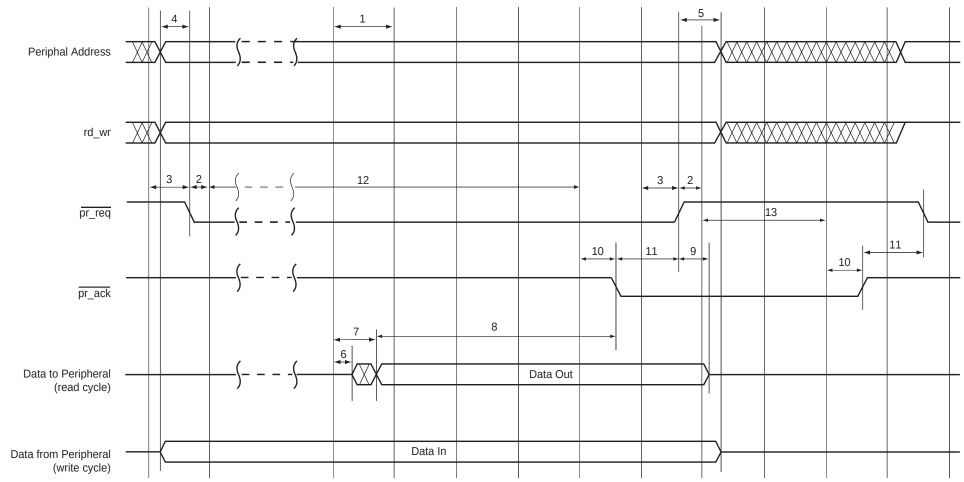


Figure 19-21 Shared RAM Interface, Peripheral Read and Write Cycle Timing

Number	Name	Explanation	Min	Nom	Max	Unit
1	t_{ckp}	Internal clock period	-	10	-	ns
2	t_{srqs}	pr_req setup time to clock (note 20)	2	-	-	ns
3	t_{srqh}	pr_req hold time from clock (note 20)	1	-	-	ns
4	t_{srs}	Peripheral address, peripheral data and rd_wr setup time to pr_req	0	-	-	ns
5	t_{srh}	Peripheral address, peripheral data and rd_wr hold time from pr_req	0	-	-	ns
6	t_{sroe}	Data output enable time from clock	7	-	-	ns
7	t_{srdv}	Data to peripheral valid from clock	-	-	17	ns
8	t_{srds}	Data to peripheral setup time to pr_ack	27	-	-	ns
9	t_{srdh}	Data to peripheral hold time from pr_req	1	-	-	ns
10	t_{srad}	pr_ack delay from clock	3	-	12	ns
11	t_{srap}	pr_req delay from pr_ack	0	-	-	ns
12	t_{srra}	pr_req to pr_ack latency	$12*t_{ckp}$	-	-	ns
13	t_{srri}	pr_req inactive to pr_ack inactive latency	$2*t_{ckp}$	$2*t_{ckp}$	$2*t_{ckp}$	ns

Table 19-68 Shared RAM Interface, Peripheral Read and Write Cycle Timing

Note 20: The pr_req signal is synchronized internally. Setup and hold times need to be taken into consideration only if detection in a specific clock cycle is required.

Shared RAM Interface, a_sel Timing

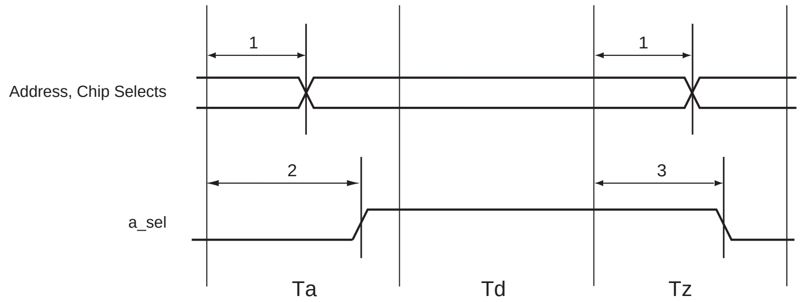


Figure 19-22 Shared RAM Interface, a_sel Timing

Number	Name	Explanation	Min	Nom	Max	Unit
1	t_a	Address and chip select delay from clock	2	-	8	ns
2	t_{srsh}	a_sel high delay from clock	3	-	10	ns
3	t_{srsl}	a_sel low delay from clock	3	-	11	ns

Table 19-69 Shared RAM Interface, a_sel Timing

Shared RAM Interface, Interrupt Timing

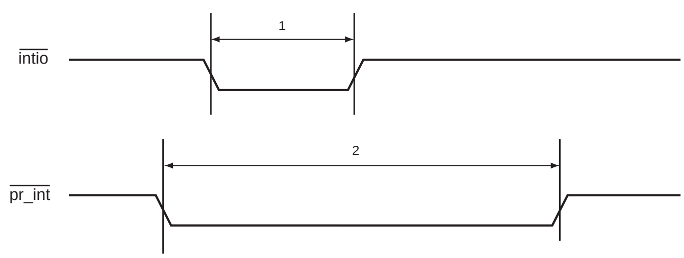


Figure 19-23 Shared RAM Interface, Interrupt Timing

Number	Name	Explanation	Min	Nom	Max	Unit
1	t_{sriw}	$\overline{\text{intio}}$ pulse width	25	-	-	ns
2	t_{srpw}	$\overline{\text{pr\_int}}$ pulse width	590	600	610	ns

Table 19-70 Shared RAM Interface, Interrupt Timing

19.13.9 Network Interface Timing

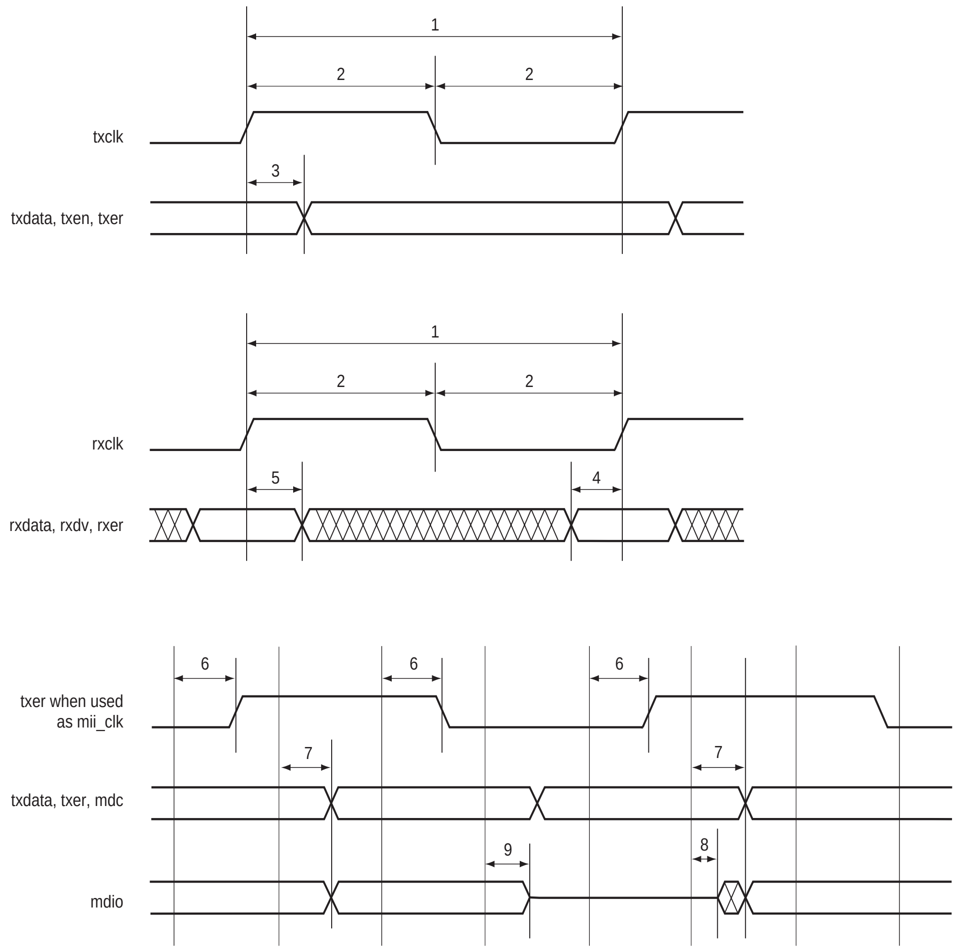


Figure 19-24 Network Interface Timing

Number	Name	Explanation	Min	Nom	Max	Unit
1	t_{etcp}	txclk and rxclk clock period, MII operation txclk and rxclk clock period, SNI operation	35 70	- -	- -	ns ns
2	t_{etcw}	txclk and rxclk pulse width	10	-	-	ns
3	t_{etdd}	txdata, txen and txer delay from txclk	4	-	16	ns
4	t_{etds}	rxdata, rxdv and rxer setup time to rxclk	3	-	-	ns
5	t_{etdh}	rxdata, rxdv and rxer hold time from rxclk	1	-	-	ns
6	t_{etcd}	txer delay from internal clock, when used as 25MHz clock output	7	-	16	ns
7	t_{etmd}	txdata, txer, mdc and mdio delay from internal clock, when controlled by the R_NETWORK_MGM_CTRL mode register	7	-	15	ns
8	t_{etme}	mdio output enable from internal clock	6	-	-	ns
9	t_{etmz}	mdio turn off time from internal clock	6	-	14	ns

Table 19-71 Network Interface Timing

19.13.10 Reset and Clock Timing

System Clock Timing

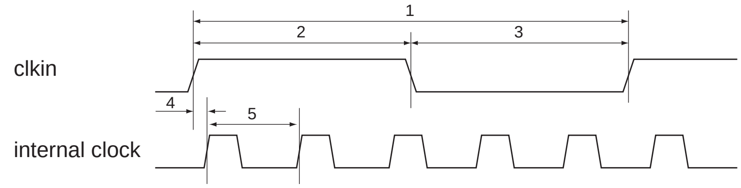


Figure 19-25 System Clock Timing Diagram

No	Name	Explanation	Min	Nom	Max	Unit
1	t_{clkp}	Input clock period (Note 21).	49	50	51	ns
2	t_{clkh}	Input clock high time.	15	-	-	ns
3	t_{clkl}	Input clock low time.	15	-	-	ns
4	t_{clkd}	Input clock to internal clock delay.	0.9	2.5	4	ns
5	t_{ckp}	Internal clock period.	-	$0.2 \cdot t_{clkp}$	-	ns

Table 19-72 System Clock Timing

Note 21: Some applications may require less tolerance on the clock period. For example, if txer is configured as the clock for Fast Ethernet, greater clock cycle accuracy is needed (see 100BASE-T standard: IEEE 802.3u.).

Reset Timing

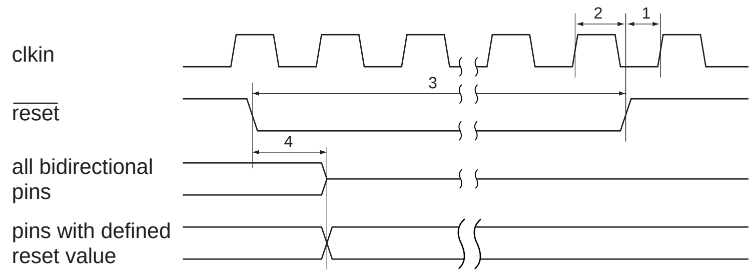


Figure 19-26 Reset Timing Diagram

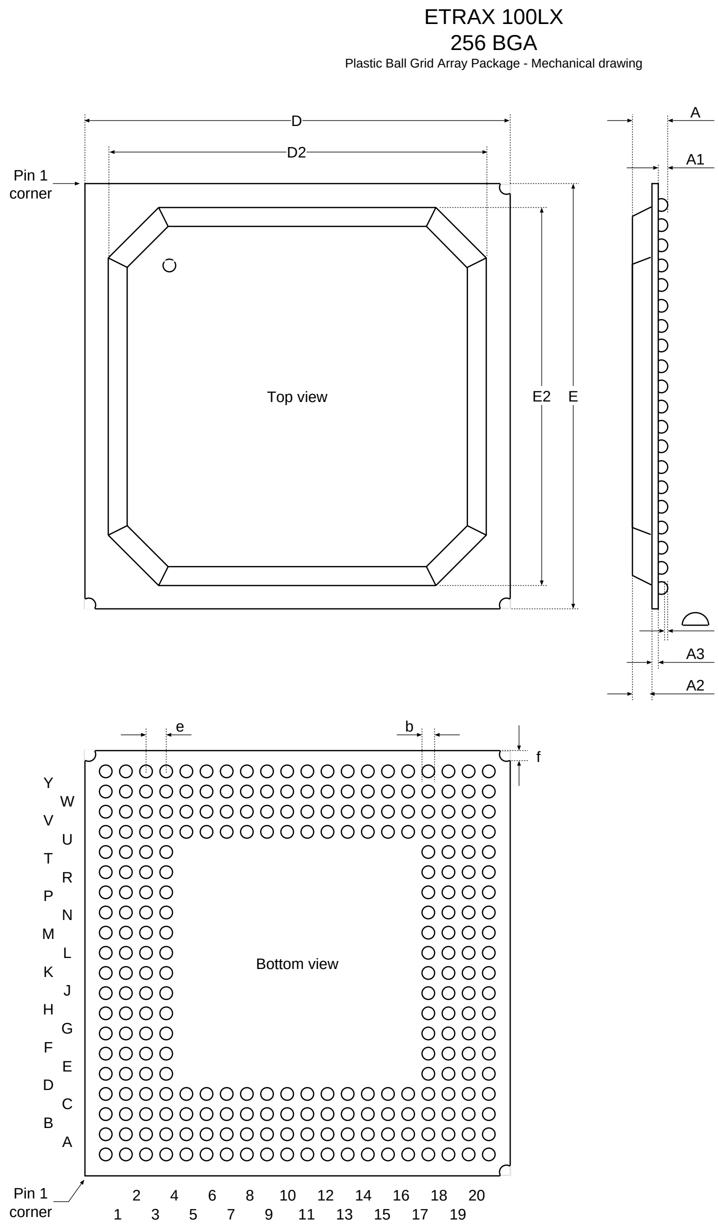
No	Name	Explanation	Min	Nom	Max	Unit
1	t_{ress}	$\overline{\text{reset}}$ setup to clkin. (Note 22).	2	-	-	ns
2	t_{resh}	$\overline{\text{reset}}$ hold from clkin. (Note 22).	1	-	-	ns
3	t_{resw}	$\overline{\text{reset}}$ pulse width.	$5 \cdot t_{clkp}$	-	-	ns
4	t_{resd}	$\overline{\text{reset}}$ to output delay.	-	-	25	ns

Table 19-73 Reset Timing

Note 22: The $\overline{\text{reset}}$ signal is internally synchronized. Setup and hold times can be ignored unless recognition on a specific clock cycle is required.

19.14 Physical Dimensions

The package of the ETRAX 100LX is a 256 lead Plastic Ball Grid Array (PBGA).



Dimension		mm
A	Min	1.92
	Nom	2.13
	Max	2.32
A1	Min	0.50
	Nom	0.60
	Max	0.70
A2	Min	1.12
	Nom	1.17
	Max	1.22
A3	Min	0.31
	Nom	0.36
	Max	0.40
⌓	Max	0.15
b	Min	0.60
	Nom	0.76
	Max	0.90
D	Min	26.90
	Nom	27.00
	Max	27.10
D2	Min	23.95
	Nom	24.00
	Max	24.35
E	Min	26.90
	Nom	27.00
	Max	27.10
E2	Min	23.95
	Nom	24.00
	Max	24.35
e	Nom	1.27
f	Nom	0.50

Figure 19-27 256 PBGA - Mechanical Drawing

Appendix A Register Address Index

This index lists the addresses used in the ETRAX 100LX, and the registers that are located at each address along with the read/write status for each register in parentheses. The final number to the right of each parentheses is the page number for that register in chapter 19.

Numerics

Appendix B PLL Clock Generation

The ETRAX 100LX uses a *Phase Locked Loop* (PLL) to generate the internal reference clock from a clock supplied to the clkin pin. The PLL circuit multiplies the clkin frequency by a factor of 15. For normal operation the clkin frequency should be 20 MHz, resulting in an internal reference frequency of 300 MHz.

PLL Pins

The following pins are used permanently by the PLL:

Name	Direction	Description
clkin	input	External clock input
plllp2	in/out	Loop filter
pllagn	output	Loop filter ground connection

Table B-1 Permanent PLL pins

The following pins are used to choose the PLL operating modes (See “Operating Modes”):

Name	Direction	Description
$\overline{\text{test}}$	input	
$\overline{\text{nmi}}$	input	

Table B-2 PLL operating modes pins

Note 1: PLL only uses the $\overline{\text{test}}$ and $\overline{\text{nmi}}$ pins during reset.

External Loop Filter

The loop filter resistor and capacitors should be connected as close as possible to the chip. Figure B-1 shows a diagram of the external loop filter, and table B-3 lists its minimum, recommended and maximum values.

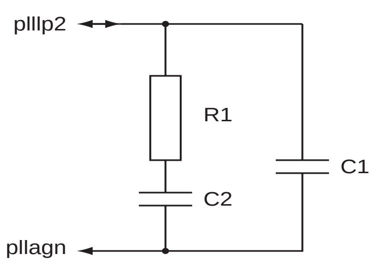


Figure B-1 External Loop Filter

	Minimum	Recommended	Maximum
R1	8.2 Ω	8.2 Ω	8.2 kΩ
C1	39pF	39pF	39pF
C2	680 pF	680 pF	1.0 µF

Table B-3 External loop filter

Operating Modes

The PLL operates in three modes:

- Normal Mode
- PLL Bypass Mode
- PLL Test Mode

Mode	test (time a)	test (time b)	nmi (time a)	nmi (time b)
Normal Mode	1	1	Don't care	Don't care
PLL Bypass Mode	0	1	0	1
PLL-test Mode	0	0	0	0

Table B-4 Operating modes

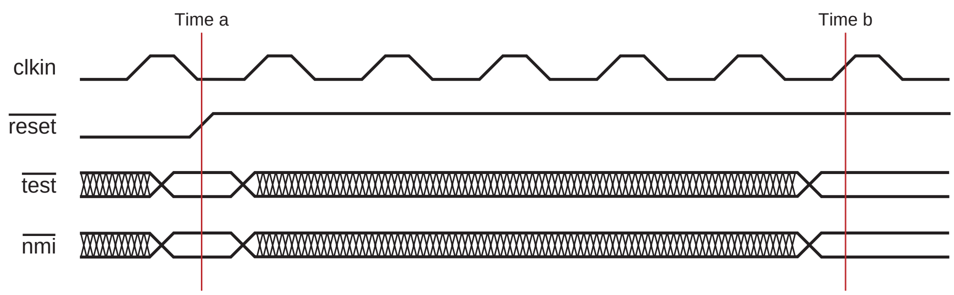


Figure B-2 PLL Mode Timing

Normal Mode

In normal mode the PLL generates the internal reference clock from the clock supplied to the **clkin** pin. The PLL circuit multiplies the clkin frequency by a factor of 15. At reset, the internal reference clock is disabled as long as **reset** is active. When **reset** is released, the reference clock is disabled for an additional 1.6 ms. During this time the PLL locks the internal reference signal. The internal 300 MHz clock is locked to the rising edge of **clkin**.

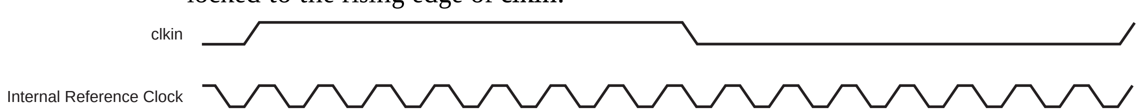


Figure B-3 Normal Mode

PLL Bypass Mode

In PLL bypass mode, the PLL is bypassed and **clkin** is used directly as the reference clock. When **reset** is active, the internal reference clock is disabled. When **reset** is released, **clkin** is immediately used as the reference clock. In this mode, the output of the 1.6 ms PLL lock timer can be read in the **pll_lock_tm** field of the R_BUS_STATUS register, which is used to test the timer.

PLL-test Mode

The PLL-test mode is only used for factory testing of the PLL. In this mode, the clock will be turned off for the rest of the chip. For more information please contact Axis Communications.