TinyRISC®
EV4101 Microprocessor
Reference Device
Technical Summary

Contents

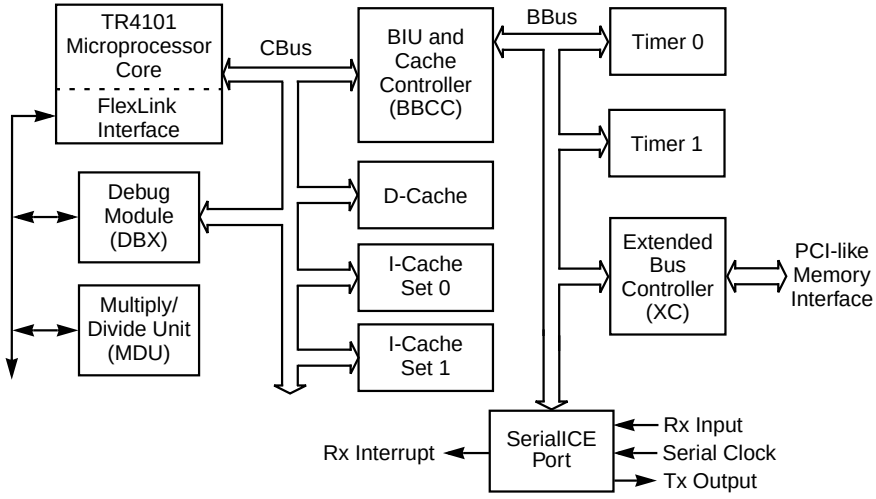
Figures

Tables

1 Introduction

The TinyRISC EV4101 Microprocessor Reference Device is a single chip implementation of the TinyRISC TR4101 Microprocessor, a component of LSI Logic's CoreWare® Library. The EV4101 Reference Device design contains an extended bus controller (XC) module and several building blocks: instruction and data caches, a timer, a multiply divide unit (MDU), and a basic bus interface unit and cache controller (BBCC). The EV4101 also includes a hardware debug (DBX) module and SerialICE™ port to enable the user to debug and download software applications through the SerialICE concept. The EV4101 is a 32-bit microprocessor that executes MIPS I, a subset of MIPS II, and a subset of the MIPS16 instructions. MIPS16 is a 16-bit wide instruction set that greatly reduces code size and thereby the amount of memory needed. Figure 1 is a block diagram of the EV4101 Reference Device.

Figure 1 EV4101 Reference Device Block Diagram



2 Features

- Includes the TR4101 microprocessor core, XC module, and BBCC, MDU, timer, DBX module, and SerialICE port building blocks
- MIPS I, MIPS II, and MIPS16 instruction set implementation
- Low power management features
- 16 Kbyte two-way set-associative instruction cache (I-cache)
- 8 Kbyte direct-mapped data cache (D-cache)
- Four-deep write buffer
- Hardware and application software debug support with SerialICE
- 3.3-volt, 0.29-micron, cell-based G10®-p CMOS process technology
- Cache controller offers advanced instruction and data streaming
- Load scheduling support
- Two 16-bit programmable timers
- Basic PCI-like 32-bit memory interface
- Full- or half-speed bus clock options
- Dual bus arbitration configurations for either host or nonhost operation
- 66 MHz at wccom, when executing MIPS I, MIPS II, and MIPS16
- 3.3-volt operation, 5-volt tolerant inputs
- 66 MIPS peak performance and 53 MIPS sustained at 66 MHz

3 Block Diagram

This section describes the internal blocks in the EV4101 Reference Device, shown in Figure 1.

The TR4101 microprocessor core is the main component of the EV4101 Reference Device. The EV4101 executes all MIPS II 32-bit based instructions except for the multiprocessor support instructions. In addition, it executes all but the double word MIPS16 instructions. The microprocessor is implemented with an efficient three-stage pipeline that is extended to four stages for Load Word (LW) and Store Word (SW) type

instructions. Standard MIPS-compiled code will run without change on the TR4101. Code containing a mixture of MIPS I/II and MIPS16 executes without any prior setup of the core.

The TR4101's instruction set includes a WAITI instruction that places the TR4101 into a low-power stall condition until an interrupt is received. For even greater power management, the EV4101 has an output pin that is asserted during this time period, so that the system designer can stop the entire system clock.

The MDU module is attached to the TR4101's FlexLink interface. It is a compact, medium-performance MIPS II compatible arithmetic engine. The MDU module supports 13-cycle integer multiply and multiply accumulate, and 34-cycle divide operation.

The DBX module is attached to the TR4101's FlexLink interface. The FlexLink interface allows a number of new instructions to be added to the microprocessor's default instruction set. The DBX module facilitates debug support by allowing the programmer to insert instruction and data access breakpoints.

The SerialICE port is a full-duplex, serial, receive and transmit port. It is used with the SerialICE concept for downloading application software to the TR4101 and for debugging. The SerialICE concept uses the breakpoints in the DBX module when required in the debug session.

The BBCC is connected to the microprocessor using the CPU bus (CBus). The BBCC is a basic bus interface unit that handles instruction/data block refilling, cache control, load scheduling, and streaming. It contains the four-deep write buffer. Connected to the BBCC are a 16 Kbyte two-way set-associative I-cache and an 8 Kbyte direct-mapped D-cache. The BBCC has an advanced method of streaming data to the microprocessor. Anytime the cache is being refilled, the microprocessor can use the data during the same cycle it is written to the cache. The BBCC is programmable using a system configuration register.

There are two internal 16-bit timers that are included in the EV4101. They are connected to the BBCC using the BBus interface. Each timer has an output that can be either toggled or pulsed. Timer 1 can be placed in a bus Watchdog Mode where a bus error is signalled on excessively long transactions. When the Timer 1 is in Watchdog Mode, the EV4101 detects a bus error when a transaction exceeds the count length that is programmed into Timer 1.

The XC module handles transactions on the BBus and translates them to a PCI-like 32-bit memory interface. Where the BBus uses separate address and data buses, the EV4101 memory interface has a multiplexed address/data bus. The XC can run the memory interface at full speed, or it can divide the clock by two for a half speed system bus. The PCI bus is a multimaster bus, as is the BBus. Therefore, many of the EV4101 bus pins are bidirectional. A PCI bus master outside the EV4101 is also a master on the BBus.

4 EV4101 Programming Model Details

This section describes the programming model for the EV4101. It describes the programming models for the TR4101, DBX module, MDU, BBCC, and the timers. This document does not describe all the EV4101 registers. The EV4101 uses an R3000-type CP0. Refer to R3000 documents for information on the EV4101 CP0 registers. For specific information on registers resident in the TR4101, refer to the *TinyRISC TR4101 Microprocessor Core Technical Manual*, document number DB14-000059-00. For further information on the BBCC, MDU, DBX module, and SerialICE port refer to the *TinyRISC TR4101 Building Blocks Technical Manual,* document number DB14-000060-00.

4.1 DBX Overview

The DBX module provides debug support in the EV4101 by providing access in real time to instruction and data access breakpoints. The DBX module is controlled by software through registers.

The DBX gives the programmer access to the DBX registers. For details on the DBX and FlexLink Interface, refer to the *TinyRISC TR4101 Microprocessor Core Technical Manual.*

To detect data and instruction addresses, the DBX monitors the CBus. Refer to the *TinyRISC TR4101 Microprocessor Core Technical Manual* for information on the CBus.

Software controls the DBX module through two new instructions, MTD (Move To Debugger) and MFD (Move From Debugger), that access the DBX registers. This software interface enables use of internal breakpoints to generate a software break to the exception handler.

For information on DBX interfaces to other elements, such as the TR4101, BBCC, and system logic, refer to the *TinyRISC TR4101 Building Blocks Technical Manual,* document number DB14-000060-00.

4.1.1 Registers

Programmers access the DBX registers by using the MTD and MFD instructions. See Section 4.1.2, “Instructions.” All bits are read/write, except for the Reserved bits, which are hard wired to zero. Software should, however, always write zeros to the Reserved bits.

The registers in this section have a register number for each Debug Register (dr) which are indicated in parentheses in the subsection headings.

DCS Register (7) – The Debug Control and Status (DCS) Register contains the status and enable bits for the debug facilities. All status bits (bits 4 through 0) are sticky, and debug events only set the bits if the enable signals are set. The bits must be cleared by using an MTD Instruction to update the DCS Register. The settings of the UD and KD bits are always the same—setting either bit sets both bits. Reset clears the DE and IBD bits. All other bits are unknown.

MTD Instructions have higher priority than status updates. The DBX updates the DCS Register with MTD data if an MTD Instruction occurs simultaneously with a status update caused by a break event.

Table 1 shows the format of the DCS Register.

Table 1 DCS Register

31	30	29	28	27	26	25	24	23	22	21 … 5	4	3	2	1	0
TR	UD	KD	Res	DW	DR	DAE	PCE	DE	IBD	Reserved	W	R	DA	PC	DB

TR **Trap Enable** **31**

Setting TR causes debug events to trap to the exception vector provided that internal breaking is enabled; that is, IBD is cleared. When TR is cleared, no trapping occurs, but debug status bits are still updated with debug event information.

UD	**User Mode Debug Event Detection Enable** Setting UD enables debug breaks in User Mode. UD and KD always contain the same value. Setting either bit sets both bits.	**30**
KD	**Kernel Mode Debug Event Detection Enable** Setting KD enables debug breaks in Kernel Mode. UD and KD always contain the same value. Setting either bit sets both bits.	**29**
Res	**Reserved Bits** These bits are hard wired to zero. Software should always write zeros to these bits.	**28, [21:5]**
DW	**Data Write Enable** If DAE is enabled, setting DW enables data write event detection.	**27**
DR	**Data Read Enable** If DAE is enabled, setting DR enables data read event detection.	**26**
DAE	**Data Access Breakpoint Enable** Setting DAE enables data address breakpoint debug events.	**25**
PCE	**Program Counter Breakpoint Enable** Setting PCE enables program counter breakpoint debug events.	**24**
DE	**Debug Enable** Setting DE enables the debugger. Clearing DE disables the debugger.	**23**
IBD	**Internal Break Disable** Setting IBD disables internal breaks. Clearing IBD enables internal breaks. When IBD is set, the DBX does not assert the DBREAK_BBEP signal on a debug event.	**22**
W	**Write Reference Status** The DBX sets this bit when it has detected a data write debug condition. This bit can only be cleared by software.	**4**
R	**Read Reference Status** The DBX sets this bit when it has detected a data read-debug condition. This bit can only be cleared by software.	**3**

DA **Data Access Debug Condition Status** **2**
The DBX sets this bit when it has detected a data access (read or write) debug condition. This bit can only be cleared by software.

PC **Program Counter Debug Condition Status** **1**
The DBX sets this bit when it has detected a program counter debug condition. This bit can only be cleared by software.

DB **Debug Condition Status** **0**
The DBX sets this bit when it has detected a debug event. This bit can only be cleared by software.

Table 2 shows how to enable the DBX functions.

Table 2 Enabling Debug Functions

TR	UD	KD	DE	IBD	Action on Debug Event
0	0	0	1	0	Status update (only)
1	1	1	1	0	Status update and internal break (trap)
x	x	x	0	x	None (DBX disabled)

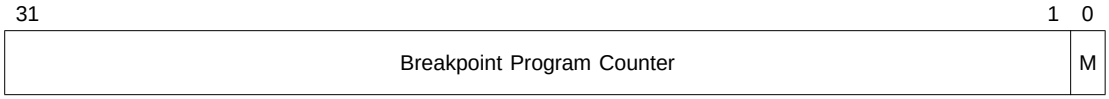
BPC Register (18) – Software uses the Breakpoint Program Counter (BPC) Register to specify a program counter breakpoint. This register is used in conjunction with the Breakpoint Program Counter Mask (BPCM) Register described later. A breakpoint is detected for any instruction in which all enabled bits in the BPC Register match the corresponding bits in the instruction fetch address. The BPC Register contents are undefined after reset.

Note that a breakpoint placed off a word boundary in MIPS II mode, or a breakpoint placed at the start of a 32-bit MIPS16 instruction will not be detected. If a break is to be generated on a 32-bit MIPS16 instruction (EXTEND or JAL/JALX), the breakpoint must be placed at the second halfword of the instruction.

The M-bit of the BPC Register is used to specify whether bits 31 to 1 of the BPC Register specifies a MIPS II address or a MIPS16 address, if M is 1 then BPC specifies a MIPS16 address; otherwise, BPC specifies a MIPS II address.

Table 3 shows the format of the BPC Register.

Table 3 BPC Register

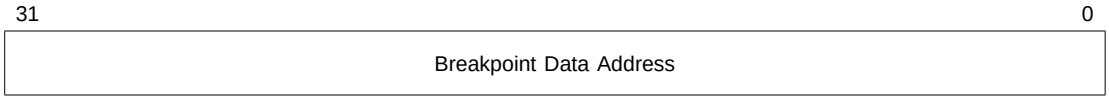


BDA Register (19) – Software uses the Breakpoint Data Address (BDA) Register to specify a data address breakpoint. This register is used in conjunction with the Breakpoint Data Address Mask (BDAM) Register described later. A breakpoint is detected for any data reference, including byte and halfword references in which all enabled bits in the BDA Register match the corresponding bits in the data address. The BDA Register contents are undefined after reset.

Note that it is the start address of a data element that is used in the detection; a data breakpoint placed within a word will not be detected unless the address is explicitly targeted by, for example, an LB instruction. If, however, bit 1 and bit 0 of the BDAM Register are both 0, then all accesses to the word are detected.

Table 4 shows the format of the BDA Register.

Table 4 BDA Register

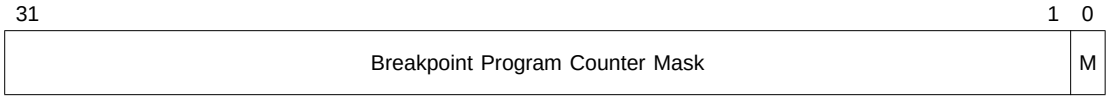


BPCM Register (20) – The Breakpoint Program Counter Mask (BPCM) Register masks bits in the BPC Register. Writing a 1 to bit *n* of the BPCM Register enables the bitwise comparison of bit n in the BPC Register with bit n in the instruction fetch address. Conversely, writing a 0 to bit n of the BPCM Register disables the comparison, and forces the breakpoint logic to assume a match between bit n in the BPC and bit n in the instruction fetch address, regardless of the true result. The BPCM Register contents are undefined after reset.

Likewise, the M-bit of the BPCM Register enables or disables the bitwise comparison of the M-bit in the BPC Register with the instruction fetch mode.

Table 5 shows the format of the BPCM Register.

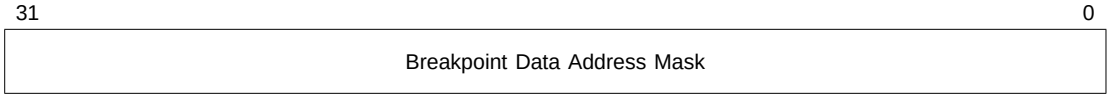
Table 5 BPCM Register



BDAM Register (21) – The Breakpoint Data Address Mask (BDAM) Register masks bits in the BDA Register. Writing a 1 to bit n of the BDAM Register enables the bitwise comparison of bit n in the BDA Register with bit n in the data reference address. Conversely, writing a 0 to bit n of the BDAM Register disables the comparison, and forces the breakpoint logic to assume a match between bit n in the BDA and bit n in the data reference address regardless of the true result. The BDAM Register contents are undefined after reset.

Table 6 shows the format of the BDAM Register.

Table 6 BDAM Register



4.1.2 Instructions

The MFD and MTD instructions access the DBX registers. These instructions use the FlexLink interface of the TR4101 and are not supported by compilers or assemblers. The following macros may be used to enable the MFD and MTD instructions:

```
#define MFD(rt,dr).word0x7C000000 + (rt << 16) + (dr << 11)
#define MTD(rs,dr).word0x78000000 + (rs << 21) + (dr << 11)
```

MFD Instruction – The Move From Debug (MFD) Instruction loads the contents of DBX Register `dr` into a General Register `rt`. This instruction is only valid if `dr` = 7, 18, 19, 20, or 21. All other `dr` values are undefined. The values for each Debug Register `dr` are indicated in parentheses in the subsection headings.

Operation: T: GPR[rt] <- DEBUG[dr]

Exceptions: None

Table 7 shows the format of the MFD Instruction. The MFD Instruction is not available from MIPS16 code.

Table 7 MFD Instruction (MIPS II format)

31 26	25 21	20 16	15 11	10 0
MFD = 011111_2	rs = 00000_2	rt	dr	0

MTD Instruction – The Move To Debug (MTD) Instruction loads the contents of General Register `rs` into a DBX Register `dr`. This instruction is only valid if `dr` = 7, 18, 19, 20, or 21. All other `dr` values are undefined.

Operation: T: DEBUG[dr] <- GPR[rs]

Exceptions: None

Table 8 shows the format of the MTD Instruction. The MTD Instruction is not available from MIPS16 code.

Table 8 MTD Instruction (MIPS II format)

31 26	25 21	20 16	15 11	10 0
MTD = 011110_2	rs	rt = 00000_2	dr	0

4.1.3 Breakpoints

This section describes the operation of the DBX Building Block in more detail.

When a breakpoint occurs, the DBX sets the proper status bit. All status bits are sticky—they are not cleared by hardware, and can only be cleared by software.

Breakpoints are only detected if the TR4101 actually executes the breaking instruction, thus breakpoints can be placed freely in the code. For example, a breakpoint placed at an instruction following any MIPS16 branch will only be detected if the branch is not taken, as one would expect, from a programmer's point of view. The general rule is that breakpoints are only reported for instructions that complete their execute stage, or for instructions that would have completed the stage were it not for the Bus Error caused by the DBX.

In addition, breakpoints are reported in the order they occur in the code. A data breakpoint followed by an instruction breakpoint results in the

data breakpoint being reported first, even though the instruction fetch address following the data load/store transaction actually appears on the CBus first.

Data Write Breakpoint – A data write breakpoint occurs when the TR4101 writes data to an address that matches all the bits of the BDA Register not masked by the BDAM Register.

The DE, DW, and DAE bits of the DCS Register enable data write breakpoints. The BDA and BDAM registers must be properly loaded before the breakpoint is enabled. When the data write breakpoint occurs and the IBD and TR bits of the DCS Register are enabled, the TR4101 signals a DBus Error. The EPC Register then points to the offending write instruction unless the write instruction was executed in a delay slot. In this case, EPC points to the jump or branch instruction instead. The data write instruction and all previous instructions complete execution.

Data Read Breakpoint – A data read breakpoint occurs when the TR4101 reads data (not an instruction fetch) from an address that matches all of the bits of the BDA Register not masked by the BDAM Register.

The DE, DR, and DAE bits of the DCS Register enable data read breakpoints. The BDA and BDAM registers must be properly loaded before the breakpoint is enabled. When the data read breakpoint occurs and the IBD and TR bits of the DCS Register are enabled, the TR4101 signals a DBus Error. The EPC Register then points to the offending read instruction unless the read instruction was executed in a delay slot. In this case, EPC points to the jump or branch instruction instead. All previous instructions complete execution. The data read instruction is completed if the data is not contained in cache, and is not completed if the data is contained in cache.

Program Counter Breakpoint – A program counter breakpoint occurs when the TR4101 is about to execute an instruction whose address and mode matches all the bits of the BPC Register not masked by the BPCM Register.

The DE and PCE bits of the DCS Register enable this breakpoint. The BPC and BPCM registers must be properly loaded before the breakpoint is enabled. If the IBD and TR bits of the DCS Register are enabled, the DBX signals an IBus Error when the TR4101 is about to execute the offending instruction. This causes a trap to the exception handler. The EPC Register then points to the instruction unless the instruction was about to be executed in a delay slot. In this case, the EPC Register points to the jump or branch instruction instead. The offending instruction is killed, but all previous instructions complete execution.

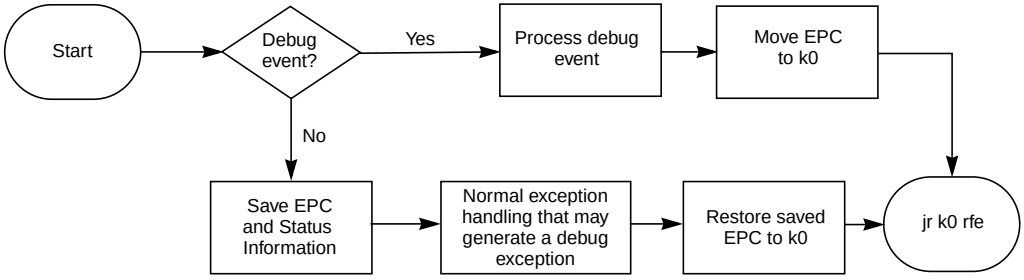
A breakpoint placed off a word boundary in MIPS II mode or a breakpoint placed at the start of a 32-bit MIPS16 instruction will not be detected. If a break is to be generated on a 32-bit MIPS16 instruction (EXTEND or JAL/JALX), the breakpoint must be placed at the second halfword of the instruction.

Note that if the M-bit of BPCM is 1, then the M-bit of BPC must be 0 in order to detect breakpoints in MIPS II mode, and the M-bit of BPC must be 1 in order to detect breakpoints in MIPS16 mode. The M-bit of BPC and BPCM can be programmed to detect, for example, all MIPS16 instructions (BPC==BPCM==0x00000001).

Breakpoint Placement – When the internal trap mechanism is enabled, with TR and IBD both enabled in the DCS Register, special care needs to be taken when using program counter breakpoints and/or data breakpoints from inside the exception handler.

If there is a possibility of triggering a debug exception from within the exception handler, the user must ensure that the necessary machine state is saved before a debug exception might occur. This can be done by saving EPC (Exception Program Counter) contents and other important status information on entry, as shown in Figure 2. However, the user must see that no debug exception occurs in critical regions such as those that process debug events, save/restore EPC contents, and so forth.

Figure 2 Typical Exception Handler Flow



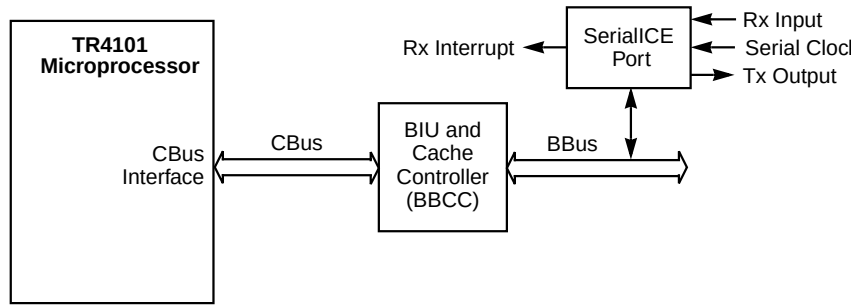
Note: jr k0 rfe are MIPS assembly language instructions typically used to return from an exception handler

4.2 SerialICE Port Overview

The SerialICE port is a full-duplex, serial, receive and transmit port. It is used with the SerialICE concept for downloading application software to the TR4101 and for debugging the TR4101. The SerialICE port is designed to work with the ICEcontroller at baud rates up to 1 M baud, providing 800 kbits of data per second. Please refer to *MIPS SerialICE Debugging Environment User's Guide* for additional information on the SerialICE concept.

Figure 3 shows a block diagram of a TR4101 system using the SerialICE port, where the TR4101 interfaces to the SerialICE port through the BBCC using the CBus, and the BBCC interfaces to the SerialICE port using the BBus.

Figure 3 TR4101 System with SerialICE Port



The SerialICE port provides the following features:

- Full-duplex operation, fixed for 8 data bits, 1 stop bit, and no parity bit.
- Clock at 16 times the bit rate to define the receiving (Rx) and transmitting (Tx) rates. The clock is common for Rx and Tx, and may either be an external clock or be generated internally from the system clock.
- An Rx ready signal to indicate that one byte of data has been received and is in the one-byte input buffer.
- Separate status and data registers for Rx and Tx. The Rx Status Register contains one bit that indicates the received data is in the SerialICE port, and one bit that indicates an overrun in the Rx input buffer. The Tx Status Register contains one bit that indicates the SerialICE port is ready to transmit data.
- Interrupt possible when new data is received.
- Serial-receive and clock-input lines that require no active signal when the SerialICE port is unused.

4.2.1 Registers

All SerialICE port registers are memory mapped, as shown in Table 9. The base address for the SerialICE port register block is in the uncached *kseg1*, and is set to 0xBFFF0200.

Table 9 SerialICE Port Register Addresses

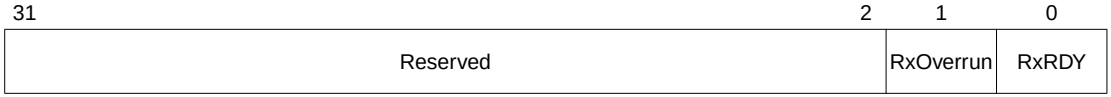
Address	Access	Register
0xBFFF0200	Read	Receive (Rx) Status Register
0xBFFF0200	Write	Receive (Rx) Setup Register
0xBFFF0204	Read	Receive (Rx) Data Register
0xBFFF0208	Read	Transmit (Tx) Status Register
0xBFFF020C	Write	Transmit (Tx) Data Register

All register transactions are read transactions with 0 in bits [31:8], and data in bits [7:0]. The register bits are mapped with bit 31 to BDATAP[31], bit 30 to BDATAP[30], and so forth. Write transactions to read-only registers are ignored, and read transactions from write-only registers return undefined data.

All registers must be accessed using word accesses only, to avoid conflict between big endian and little endian data structures, and to avoid partial update problems.

Receive (Rx) Status Register – The Rx Status Register, shown in Table 10, provides status information for the Rx operation, and indicates the validity of the Rx Data Register. The register is a read-only register.

Table 10 Rx Status Register



Reserved **Reserved Bits** **[31:2]**
These bits are reserved. They are read as zeros.

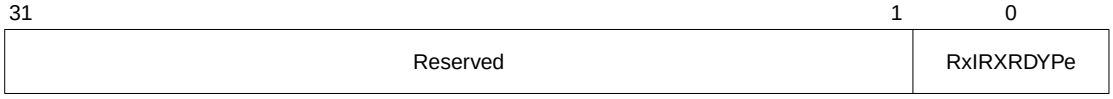
RxOverrun **Rx Overrun** **1**
This bit is set at a valid Stop bit when a new byte is received before the previously received byte has been read, as indicated by the RxRDY bit being set. The newly received byte is discarded, and the pending byte indicated by the RxRDY bit, is retained.

When the RxOverrun bit is set, it indicates that at least one byte in the serial input stream is lost. The bit is cleared when the Rx Status Register is read. This ensures that an overrun that occurs between a read of the Rx Status Register and a read of the Rx Data Register is indicated the next time the Rx Status Register is read. The RxOverrun bit is cleared at reset.

RxRDY **Rx Byte Ready** **0**
This bit is set to 1 when a byte is received by the Rx block. It is cleared to 0 when the Rx Data Register is read, and at reset. When the bit is enabled by the RxIRXRDYPe bit in the Rx Setup Register, it is connected to Interrupt 2 in a logical OR with the external interrupt request on INT2.

Receive (Rx) Setup Register – The Rx Setup Register, shown in Table 11, enables or disables the use of the RxRDY bit in the Rx Status Register, for possible interrupt on Interrupt 2. The register is a write-only register.

Table 11 Rx Setup Register

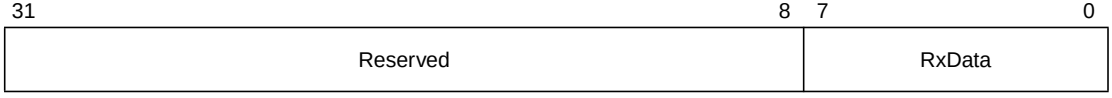


Reserved **Reserved Bits** **[31:1]**
These bits are reserved. Write transactions to these bits are ignored.

RxIRXRDYPe **Rx IRXRDYP Enable** **0**
When this bit is set to 1, the RxRDY bit in the Rx Status Register is connected to Interrupt 2 in a logical OR with the external interrupt request on INT2. When the bit is cleared to 0, only the external INT2 drives the Interrupt 2 on the CPU. The bit is cleared at reset.

Receive (Rx) Data Register – The Rx Data Register, shown in Table 12, holds received data. The register is undefined after reset. The register is a read-only register.

Table 12 Rx Data Register

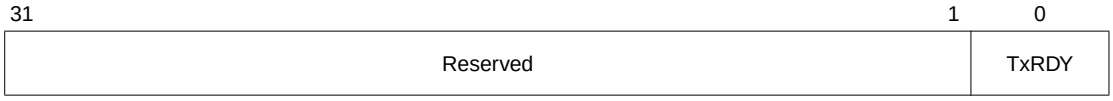


Reserved **Reserved Bits** **[31:8]**
These bits are reserved. They are read as zeros.

RxData **Received Bit Stream** **[7:0]**
This field holds data received from the ICERXP serial input signal. Data is valid only when the RxRDY bit in the Rx Status Register is set. The register is undefined after reset.

Transmit (Tx) Status Register – The Tx Status register, shown in Table 13, provides status information for the Tx operation. The register is a read-only register.

Table 13 Tx Status Register

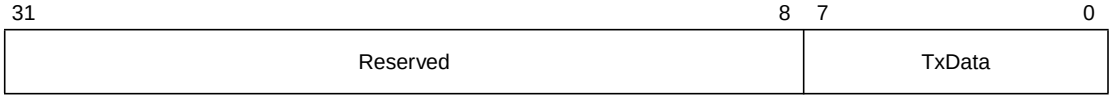


Reserved **Reserved Bits** **[31:1]**
These bits are reserved. They are read as zeros.

TxRDY **Tx Ready** **0**
This bit is set to 1 when the byte in the Tx Data Register has been transmitted and the Tx Data Register is ready for the next byte. The bit is cleared to 0 by a write to the Tx Data Register. The bit is set to 1 at reset.

Transmit (Tx) Data Register – The Tx Data Register, shown in Table 14, holds transmission data. The register is a write-only register.

Table 14 Tx Data Register



Reserved **Reserved Bits** **[31:8]**
These bits are reserved. Write transactions to these bits are ignored.

TxData **Transmitted Bit Stream** **[7:0]**
The Tx Data Register may be written with data to be transmitted on ICETXP, provided that the TxRDY bit in the Tx Status Register is set. A write to the register is ignored when the TxRDY bit is cleared.

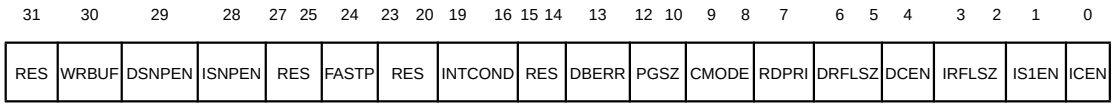
4.3 BBCC Overview

The EV4101 has a 16 Kbyte two-way set associative I-cache. Each I-cache set (8 Kbyte each) contains 512 four-word lines. The EV4101 also has a 8 Kbyte D-cache, which also consists of 512 four-word lines. Each line in the cache has a 19-bit tag and four valid bits. The D-cache is write-through. Each cache set can be individually enabled or disabled by setting the ICEN, IS1EN, and DCEN control bits of the System Configuration Register (SCR).

4.4 System Configuration Register

An SCR exists at 0xBFFF.0000. All bits are reset to zero at power-up. All bits are readable and writable. Bit 13 (DBERR) is sticky, which means that once set, it can only be reset by software.

Table 15 Configuration Register



RES **Reserved Bits** **31, [27:25], [23:20], [15:14]**
These bits are reserved. For future compatibility, always write zero to these bits.

WRBUF **Write Buffer Depth Control** **30**
When this bit is set to one, the write buffer is four deep. When this bit is zero, the write buffer is one deep.

DSNPEN **Data Cache Snoop Enable** **29**
When this bit is set to one, Data Cache snooping is enabled when the EV4101 is not the bus master.

ISNPEN **Instruction Cache Snoop Enable** **28**
When this bit is set to one, Instruction Cache snooping is enabled when the EV4101 is not the bus master.

FASTP **Fast Transactions** **24**
PCI protocol requires a dead cycle on the AD bus between drivers of the AD bus. This cycle occurs after the driving of the address and the receiving of data, and again after the receiving of data on one transaction and the driving of the address for the next transaction. Setting this bit to one removes the extra cycle between transactions and between the address and data phases (the safety measure is bypassed).

INTCOND **Interrupt/CPCOND Control** **[19:16]**

This field selects whether the INT[5:2] pins control the EV4101 condition bits or interrupts.

INTCOND	Description
Bit 19	0: INT5 Controls EV4101 Condition Bit 3 1: INT5 Controls EV4101 Interrupt 5
Bit 18	0: INT4 Controls EV4101 Condition Bit 2 1: INT4 Controls EV4101 Interrupt 4
Bit 17	0: INT3 Controls EV4101 Condition Bit 1 1: INT3 Controls EV4101 Interrupt 3
Bit 16	0: INT2 Controls EV4101 Condition Bit 0 1: INT2 Controls EV4101 Interrupt 2

DBERR **Data Bus Error** **13**

A one on this bit indicates that a data bus error occurred. The bit is set until the EV4101 clears it. This bit is connected to Interrupt 0 in a logical OR with the external interrupt request on INT0.

PGSZ[2:0] **Page Size** **[12:10]**

This field determines the page size in words of write bursts.

PGSZ[2:0]	Page Size (Words)
0b000	16
0b001	32
0b010	64
0b011	128
0b100	256
0b101	512
0b110	1024
0b111	2048

CMODE[1:0] **Cache Mode** **[9:8]**

This field determines the cache mode.

CMODE[1:0]	Cache Mode
0b00	Normal
0b01	Software Test of I-Cache Data RAM
0b10	Software Test of I-Cache Tag RAM
0b11	Software Test of D-Cache Tag RAM

RDPRI **Read Priority** **7**

A one on this bit enables read priority.

DRFLSZ[1:0] **Data Refill Size** **[6:5]**

This field determines the data refill size for cacheable data fetches only.

DRFLSZ[1:0]	Data Refill Size(Words)
0b00	1
0b01	2
0b10	4
0b11	8

DCEN **D-Cache Enable** **4**

Setting this bit enables the D-cache.

IRFLSZ[1:0] **Instruction Refill Size** **[3:2]**

This field determines the instruction refill size for cacheable instruction fetches only.

IRFLSZ[1:0]	Instruction Refill Size (Words)
0b00	1
0b01	2
0b10	4
0b11	8

IS1EN **I-Cache Set 1 Enable** **1**

When ICEN is set to one, a one on this bit enables I-cache Set 1.

ICEN **I-Cache Set 0 Enable** **0**

A one on this bit enables I-cache Set 0.

4.4.1 Locking of I-Cache Lines

The I-cache Set 0 Tag RAM contains a lock bit (in addition to the valid bits) to indicate that the line is locked. This bit can be written through Software Test mode. When the bit is set, the BBCC will not replace the line in Set 0 unless the address matches the tag in the tag RAM. By using this feature, desired lines of I-cache Set 0 can be written with important instructions and then locked. Or, the lines can be first locked and fetched by the EV4101 later when the instruction is accessed.

4.4.2 Software Test Mode

Software Cache Test Mode allows the EV4101 to write and read to cache RAMs to which it would not normally have direct access: the I-cache Data RAMs and the I-cache and D-cache Tag RAMs. This mode is useful for initializing the cache RAMs on reset, or for locking code into the I-Cache Set 0.

When the SCR is written such that CMODE[1:0] indicates that the BBCC is in Software Cache Test Mode, the BBCC interprets loads and stores differently. All loads come from the appropriate cache RAM, unless the address is in the *kseg1* address space. All stores are written to the appropriate cache RAM if they are not in *kseg1*. When CMODE[1:0] indicates that the BBCC is in Software Cache Test Mode for the I-cache, the I-cache set is specified by IS1EN in the SCR.

During Software Cache Test Mode, the software test code should be in *kseg1*. The DCEN bit in the SCR must be on for Software Cache Test Mode.

Table 16 shows the settings of the System Configuration Register for the various Software Cache Test Modes.

Table 16 SCR Settings for Software Cache Test Mode

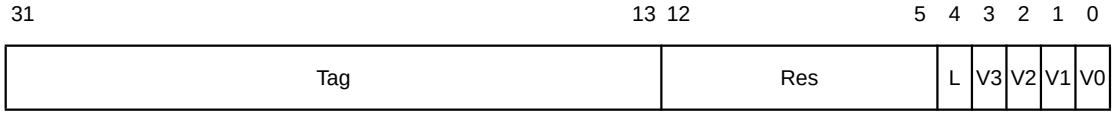
Cache Mode CMODE[1:0]	D-Cache Enable DCEN	I-Cache Set 1 Enable IS1EN	Software Cache Test Mode
01	1	0	I-Cache Set 0 Data
01	1	1	I-Cache Set 1 Data
10	1	0	I-Cache Set 0 Tag
10	1	1	I-Cache Set 1 Tag
11	1	–	D-Cache Tag

ADDR[9:0] is used to index the RAMs. For the data RAMs, the lower two bits of the address are ignored. All loads and stores are interpreted as full word transactions. For the tag RAMs, the lower four bits are ignored. Loads and stores to the tag RAMs are line operations.

During stores to the data RAM of the I-cache, the data is on DATAP[31:0]. During stores to the tag RAM of the I-cache or D-cache, the tag originates from the upper bits of ADDR[31:2] (the tag normally used for the cache). The Lock bit originates from DATAP4, and the Valid bits originate from DATAP[3:0].

Fetches from the I-cache Data RAM are straightforward. The RAM places its 32 bits of data on DATAP[31:0]. Table 17 shows data returned from reading the tag RAM of an 8 Kbyte cache while in Software Cache Test Mode.

Table 17 Data Returned from Reading Tag Ram



The following is sample code to lock six instructions in the I-cache.

```
SETUP_T0:
    li     t0, 0x000002fd # sw test to write tags to Set 0
```

```
    li     t1, 0xbfff0000 # address of Conf. Reg.
    sw     t0, 0(t1)      # store to Conf. Reg.
    lw     r0, 0(t1)      # to flush Write Buffer
    addi   r0, r0, 1      # force dependency on load

    li     t0, 0x80000000# tag
    li     t1, 0x1f       # lock, all words valid
    li     t2, 0x13       # lock, two words valid

    sw     t1, 0x80(t0) # line 0
    sw     t2, 0x90(t0) # line 1

SETUP_D0:
    li     t0, 0x000001fd# sw test to write data to Set 0
    li     t1, 0xbfff0000# address of Conf. Reg.
    sw     t0, 0(t1)      # store to Conf. Reg.
    lw     r0, 0(t1)      # to flush Write Buffer
    addi   r0, r0, 1      # force dependency on load

    li     t0, 0x80000000# tag doesn't really matter
    li     t1, 0x3c0eb000# lui t6, 0xb000
    li     t2, 0x8dcf0000# lw t7, 0(t6)
    li     t3, 0x21ef0001# addi t7, t7,1
    li     t4, 0xadcf0000# sw t7, 0(t6)
    li     t5, 0x03e00008# jr ra
    li     t6, 0x00000000# nop

    sw     t1, 0x80(t0) # Instruction 0
    sw     t2, 0x84(t0) # Instruction 1
    sw     t3, 0x88(t0) # Instruction 2
    sw     t4, 0x8c(t0) # Instruction 3
    sw     t5, 0x90(t0) # Instruction 4
    sw     t6, 0x94(t0) # Instruction 5
```

4.4.3 Block Refills

The BBCC can request a block refill if the fetch resides in cacheable space. If the address is cacheable then a block fetch is started by the BBCC starting with the desired word first. The IBLKSZ and DBLKSZ bits in the SCR selects block sizes of two, four, or eight. Instruction block size and data block size are controlled separately. When the end of the block is reached, the BBCC wraps around and requests the first word of the block until the entire block is refilled. The external memory system can terminate the block request at any time.

4.4.4 Inpage Writes

The BBCC requests back-to-back write transactions if possible. The PGSZ field of the SCR determines the page size that is used in determining whether to prioritize the next write transaction. The EV4101 asserts a Write Burst Request if a pending write is in the same page. If the Write Burst Acknowledge is received by the EV4101, the next transaction will be the inpage write. The write transaction still appears as two separate transactions (as opposed to a burst write).

4.4.5 Write Buffer

The EV4101 has a configurable depth write buffer. The WRBUF bit in the SCR configures the write buffer depth as one or four deep. If the Write Buffer depth is four, the TR4101 can execute up to four Store Word (SW) instructions and not be stalled. The SWs are held in the write buffer until the BBCC can service them.

4.4.6 Read Priority

The BBCC can be configured to prioritize Load Word (LW) instructions over a SW that is already in the write buffer. If the RDPRI bit in the SCR is one, the BBCC services a LW before a SW, even if the microprocessor requested the SW earlier.

4.4.7 Snooping

The BBCC can be configured to snoop when the EV4101 is not the master. Two control bits ISNPEN and DSNPEN control whether snooping is done by the BBCC. If ISNPEN is set, then the I-caches are snooped whenever the external master does a write transaction. Similarly, the SDNPEN controls data cache snooping. If the address of the write transaction matches the cache line, then the *entire* cache line is invalidated. The I-cache or D-cache snoop takes two cycles. Snooping both caches takes four cycles. The EV4101 asserts the Snoop Done output to indicate that the snoop is done and the transaction can continue.

4.4.8 Bus Errors

The BBCC signals Instruction bus errors *synchronously* to the microprocessor so that the microprocessor will take an instruction bus

error exception. But, because of load scheduling and the write buffer, data bus errors are not signalled to the microprocessor as bus errors. Instead, a data bus error sets an interrupt bit in the SCR (bit 13) that is set until the microprocessor clears the bit. In the EV4101, this data bus error interrupt is tied to the microprocessor interrupt 0, Cause Register bit 10.

4.5 Timer Overview

The EV4101's timer is attached internally to the BBCC's BBus. It contains two general purpose timers: Timer 0 and Timer 1. Each timer is a 16-bit down-counter with an initial count register and a current count register. A mode register defines the operation of both timers.

Both timers have a general-purpose mode where the timer counts down every cycle (internal system clock, not external bus clock). When the count reaches zero, a timer output (one for Timer 0 and one for Timer 1) will either toggle or pulse (H -> L -> H).

When enabled through the Mode Register, the timer starts from the initial count register and counts down. Writing a zero to bit 0 or bit 8 of the Mode Register disables the respective timer and causes the timer to hold its current count value. If the counter is enabled again after it is disabled, the counter restarts from the initial count. It will not count down from the current count value.

Timer 0 has an additional sticky bit. In this mode, the timer asserts the active-LOW signal when it counts to zero. The interrupt signal can be deasserted through a microprocessor write to clear the sticky bit in the Interrupt Status Register.

Timer 1 has an additional Watchdog Mode feature. In this mode, the timer starts from the initial count value and starts counting down whenever the BBCC starts a bus cycle. If the counter reaches 0, Timer 1 signals a bus error to the BBCC and terminates the transaction. If the bus cycle ends before the timer reaches 0, the timer is disabled and holds the current count value. A new bus transaction causes the timer to restart from the initial count value.

After reset, both timers are disabled, both Initial Count registers are zero, and the timer outputs are HIGH.

All timer registers are 16 bits, and memory is mapped as shown in Table 18.

Table 18 Timer Registers

Address	Register
0xBFFF.0100	Timer 0 Initial Count Register
0xBFFF.0104	Timer 0 Current Count Register
0xBFFF.0108	Timer 1 Initial Count Register
0xBFFF.010C	Timer 1 Current Count Register
0xBFFF.0110	Mode Register
0xBFFF.0114	Interrupt Status Register (Timer 0 Only)

Table 19 shows how the different bits in the Mode Register control the operation of the timers.

Table 19 Mode Register

Bit	Control	Settings
0	Timer 0 Enable	0 = Disable, 1 = Enable
1	Timer 0 Output	0 = Toggle, 1 = Pulse
8	Timer 1 Enable	0 = Disable, 1 = Enable
9	Timer 1 Output	0 = Toggle, 1 = Pulse
10	Timer 1 Mode	0 = General Purpose, 1 = Watchdog Mode[1]

1. When Timer 1 is in the Watchdog Mode, the timeout output is a bus error pulse regardless of the output mode (bit 9).

Table 20 shows the bits in the Interrupt Status Register.

Table 20 Status Register

Bit	Control	Settings
0	Sticky Bit Enable	0 = Sticky Bit Disabled, 1 = Sticky Bit Enabled
1	Sticky Bit	Set to one when Timer 0 counts to zero; microprocessor clears this bit from the exception handler.

5 Instruction Set Summary

Table 21 through Table 25 summarize the instruction sets for the TR4101 Microprocessor Core and the EV4101 Microprocessor Reference Device. This includes all MIPS I and MIPS II instructions; instructions are MIPS I unless otherwise marked. Table 26 lists all MIPS16 instructions.

Table 21 TR4101/EV4101 Instruction Set Summary (32-bits)

Load/Store Instructions			
LB	Load Byte	LWR	Load Word Right
LBU	Load Byte Unsigned	SB	Store Byte
LH	Load Halfword	SH	Store Halfword
LHU	Load Halfword Unsigned	SW	Store Word
LW	Load Word	SWL	Store Word Left
LWL	Load Word Left	SWR	Store Word Right
Arithmetic Instructions: ALU Immediate			
ADDI	Add Immediate	ORI	OR Immediate
ADDIU	Add Immediate Unsigned	SLTI	Set on Less Than Immediate
ANDI	AND Immediate	SLTIU	Set on Less Than Immediate Unsigned
LUI	Load Upper Immediate	XORI	Exclusive OR Immediate

(Sheet 1 of 3)

Table 21 TR4101/EV4101 Instruction Set Summary (32-bits) (Cont.)

Arithmetic Instructions: Three-Operand, Register-Type			
ADD	Add	SLT	Set on Less Than
ADDU	Add Unsigned	SLTU	Set on Less Than Unsigned
AND	Logical AND	SUB	Subtract
NOR	Logical NOR	SUBU	Subtract Unsigned
OR	Logical OR	XOR	Exclusive Logical OR
Jump and Branch Instructions			
BCzF	Branch on Coprocessor z False	BLTZAL	Branch on Less Than Zero and Link
BCzT	Branch on Coprocessor z True	BNE	Branch on Not Equal
BEQ	Branch on Equal	J	Jump
BGEZ	Branch on Greater Than or Equal to Zero	JAL	Jump and Link
BGEZAL	Branch on Greater Than or Equal to Zero and Link	JALR	Jump and Link Register
BGTZ	Branch on Greater Than Zero	JALX[1]	Jump and Link Exchange
BLEZ	Branch on Less Than or Equal to Zero	JR	Jump Register
BLTZ	Branch on Less Than Zero		
Branch Likely Instructions			
BCzFL[2]	Branch on Coprocessor z False Likely	BGTZL[2]	Branch on Greater Than Zero Likely
BCzTL[2]	Branch on Coprocessor z True Likely	BLEZL[2]	Branch on Less Than or Equal to Zero Likely
BEQL[2]	Branch on Equal Likely	BLTZALL[2]	Branch on Less Than Zero and Link Likely
BGEZALL[2]	Branch on Greater Than or Equal to Zero and Link Likely	BLTZL[2]	Branch on Less Than Zero Likely
BGEZL[2]	Branch on Greater Than or Equal to Zero Likely	BNEL[2]	Branch on Not Equal Likely

(Sheet 2 of 3)

Table 21 TR4101/EV4101 Instruction Set Summary (32-bits) (Cont.)

Trap Instructions			
TEQ[2]	Trap if Equal	TLT[2]	Trap if Less Than
TEQI[2]	Trap if Equal Immediate	TLTI[2]	Trap if Less Than Immediate
TGE[2]	Trap if Greater Than or Equal	TLTIU[2]	Trap if Less Than Immediate Unsigned
TGEI[2]	Trap if Greater Than or Equal Immediate	TLTU[2]	Trap if Less Than Unsigned
TGEIU[2]	Trap if Greater Than or Equal Immediate Unsigned	TNE[2]	Trap if Not Equal
TGEU[2]	Trap if Greater Than or Equal Unsigned	TNEI[2]	Trap If Not Equal Immediate
Coprocessor Instructions			
BCzF	Branch on Coprocessor z False	LWCz	Load Word to Coprocessor z ($z \neq 0$)
BCzT	Branch on Coprocessor z True	MTCz	Move to Coprocessor z
COPz	Coprocessor Operation	MFCz	Move from Coprocessor z
CTCz	Move Control to Coprocessor z	SWCz	Store Word from Coprocessor z ($z \neq 0$)
CFCz	Move Control from Coprocessor z		
Shift Instructions			
SLL	Shift Left Logical	SRAV	Shift Right Arithmetic Variable
SLLV	Shift Left Logical Variable	SRL	Shift Right Logical
SRA	Shift Right Arithmetic	SRLV	Shift Right Logical Variable
Special Control Instructions			
BREAK	Breakpoint	SYSCALL	System Call
(Sheet 3 of 3)			

1. New 32-bit MIPS instruction.
2. MIPS II instructions.

Table 22 TR4101/EV4101-Specific System Control Processor (CP0) Instructions (32-bits)

MFC0	Move from CP0	RFE	Restore from Exception
MTC0	Move to CP0	WAITI	Wait for Interrupt

Table 23 Multiply/Divide Instructions (with MDU)

MULT	Multiply	DIV	Divide
MULTU	Multiply Unsigned	DIVU	Divide Unsigned
MADD[1]	Multiply Add	MFHI	Move from HIGH
MADDU[1]	Multiply Add Unsigned	MFLO	Move from LOW
MSUB[1]	Multiply Subtract	MTHI	Move to HIGH
MSUBU[1]	Multiply Subtract Unsigned	MTLO	Move to LOW

1. New 32-bit MIPS instruction.

Table 24 DBX Instructions

MFD[1]	Move from Debug	MTD[1]	Move to Debug

1. New 32-bit MIPS instruction.

Table 25 Unimplemented MIPS II Instructions

COP1	All floating point instructions	ERET	Exception Return
LL	Load Linked Word	LDCz	Load Doubleword to Coprocessor
SC	Store Conditional Word	SDCz	Store Doubleword to Coprocessor
SYNC	Synchronize Shared Memory		

Table 26 TR4101/EV4101 Instruction Set Summary (MIPS16)

Load/Store Instructions			
LB	Load Byte	LW	Load Word
LBU	Load Byte Unsigned	SB	Store Byte
LH	Load Halfword	SH	Store Halfword
LHU	Load Halfword Unsigned	SW	Store Word
Arithmetic Instructions: ALU Immediate			
LI	Load Immediate	SLTIU	Set on Less Than Immediate Unsigned
ADDIU	Add Immediate Unsigned	CMPI	Compare Immediate
SLTI	Set on Less Than Immediate		
Arithmetic Instructions: Two/Three Operand, Register Type			
ADDU	Add Unsigned	AND	AND
SUBU	Subtract Unsigned	OR	OR
SLT	Set on Less Than	XOR	Exclusive OR
SLTU	Set on Less Than Unsigned	NOT	NOT
CMP	Compare	MOVE	Move
NEG	Negate		
Multiply/Divide Instructions (with optional MDU)			
MULT	Multiply	DIVU	Divide Unsigned
MULTU	Multiply Unsigned	MFHI	Move From HIGH
DIV	Divide	MFLO	Move From LOW
Jump and Branch Instructions			
JAL	Jump and Link	BNEZ	Branch on Not Equal to Zero
JALX	Jump and Link Exchange	BTEQZ	Branch on T Equal to Zero
JR	Jump Register	BTNEZ	Branch on T Not Equal to Zero
JALR	Jump and Link Register	B	Branch Unconditional
BEQZ	Branch on Equal to Zero		
(Sheet 1 of 2)			

Table 26 TR4101/EV4101 Instruction Set Summary (MIPS16) (Cont.)

Shift Instructions			
SLL	Shift Left Logical	SLLV	Shift Left Logical Variable
SRL	Shift Right Logical	SRLV	Shift Right Logical Variable
SRA	Shift Right Arithmetic	SRAV	Shift Right Arithmetic Variable
Special Control Instructions			
EXTEND	Extend	BREAK	Breakpoint
(Sheet 2 of 2)			

6 Signal Descriptions

This section describes the EV4101 interface. It contains the following subsections:

- Bus Interface Signals
- Miscellaneous Signals

Within each subsection, the signals are described in alphabetical order by mnemonic. Each signal definition contains the mnemonic and the full signal name. In the descriptions that follow, the verb assert means to drive TRUE or active. The verb deassert means to drive FALSE or inactive.

The direction of many of the bus pins depends on whether the EV4101 is bus master or there is an external bus master. Direction is marked as follows: im/xm = i/o, if the pin is input when the EV4101 is bus master (im = internal master) and output when there is an external bus master (xm = external master).

Similarly, the direction of the arbiter pins depends on whether the EV4101 is the bus arbiter or an external arbiter exists (see ARBMODE2P pin). Direction is marked as follows: ia/xa = i/o if the pin is input when the internal arbiter (ia) is used and output if an external arbiter (xa) is used.

6.1 Bus Interface Signals

AD[31:0] **Address/Data Bus** **Bidirectional**
AD is the multiplexed address/data for the bus transaction. This bus supplies the address during the first clock of a bus transaction, and provides the data thereafter.

ASN **Address Strobe** **Output**
ASN is active during the first clock cycle of every transaction. A LOW on ASN indicates a valid address is on AD[31:0].

CBEN[3:0] **Command/Byte Enables** **Bidirectional, im/xm=o/i**
The Command/Byte enable signals carry command information on the first cycle of a bus transaction and carry byte enables during subsequent cycles. Valid commands that the EV4101 drives on CBEN are Memory Read (0b0110), Memory Write (0b0111), and Memory Line Read (0b1110). Valid byte enables during reads and writes are listed in the following table.

CBEN[3:0]	Transfer Type
0b0000	Word
0b0001	Tribyte
0b0011	Halfword
0b0111	Byte
0b1000	Tribyte
0b1011	Byte
0b1100	Halfword
0b1101	Byte
0b1110	Byte

DEVSELN **Device Select** **Bidirectional, im/xm=i/o**
The PCI target asserts DEVSELN when it has acknowledged that it is the slave for a particular transaction. A bus error occurs if DEVSELN is not asserted within six clock cycles of the assertion of FRAMEN, resulting in an exception.

FRAMEN **Frame** **Bidirectional, im/xm=o/i**

FRAMEN is asserted LOW for at least one clock cycle at the beginning of every read and write transaction. During bursts, the EV4101 extends assertion of FRAMEN, and deasserts FRAMEN during the final data transfer.

GNTN **Bus Grant** **Bidirectional, ia/xa=o/i**

The ARBMODE2P input determines whether the EV4101 is in Arbitration Mode 1 or Arbitration Mode 2. GNTN is an output in Arbitration Mode 1, and an input in Arbitration Mode 2. In Mode 1, the EV4101 contains the bus arbiter, and asserts GNTN to indicate that an external master has control of the bus on the next transaction. (This operation follows the PCI hidden arbitration protocol, where the end of the current transaction is defined as the cycle when both FRAMEN and IRDYN are sampled deasserted.)

In Mode 2, the bus arbiter is external to the EV4101 and GNTN is an input. When the EV4101 samples the assertion of GNTN, it starts a transaction on the bus after the current transaction has been completed.

INPAGEN **Inpage Transaction** **Output**

The EV4101 asserts INPAGEN when the address of the current transaction is in the same page as the address of the previous transaction. The EV4101 must be the consecutive master of both transactions. If an external master performs a transaction between the EV4101's first and second transactions, then INPAGEN is cleared during the second transaction. INPAGEN can be monitored to enable the designer to implement DRAM Row Address Strobe (RAS) parking.

IRDYN **Initiator Data Ready** **Bidirectional, im/xm=o/i**

During writes, when asserted LOW, IRDYN indicates to the target that data is valid. During reads, when asserted LOW, IRDYN indicates the EV4101 can receive data. Data is transferred when both IRDYN and TRDYN are valid.

REQN **Bus Request** **Bidirectional, ia/xa=i/o**

The ARBMODE2P input determines whether the EV4101 is in Arbitration Mode 1 or Arbitration Mode 2. REQN is an input in Arbitration Mode 1 and an output in Arbitration Mode 2. In Mode 1, the EV4101 contains the bus arbiter

and monitors REQN to determine if an external DMA device requires mastership of the bus. In Mode 2, the bus arbiter is external to the EV4101, and the EV4101 asserts REQN when it needs to perform a transaction.

SDONEP **Snoop Done** **Output**

The EV4101 asserts the Snoop Done signal when it has completed its snoop. When a DMA transaction is being performed on the bus, the target should monitor SDONEP, and not assert TRDYN until SDONEP is HIGH. SDONEP is generated asynchronously as the logical combination of FRAMEN and CBEN[0].

STOPN **Stop Bus Transaction** **Input**

This signal is used in conjunction with TRDYN and DEVSELN to inform the initiator to either abort the transaction, retry the transaction, or stop the transaction.

- The transaction is aborted when STOPN is asserted and TRDYN and DEVSELN are both deasserted, resulting in a bus error on the BBus.
- The transaction is retried when STOPN and DEVSELN are both asserted and TRDYN is deasserted. The EV4101 usually retries the same transaction after allowing other masters to acquire the bus, if requested, but it is not guaranteed to do so. The EV4101 has the option to prioritize its transactions internally, so the initial transaction that resulted in a retry may eventually be retried, but not necessarily in the following cycle. Because a retry can occur in the middle of a cache refill, the EV4101 will not perform the transaction again, unless the instruction/data is absolutely needed.
- The transaction is stopped when STOPN, TRDYN, and DEVSELN are deasserted. The target uses this mechanism to tell the master that it cannot perform a burst, or that it wants the current data transfer in the burst to be the final data transfer. If the EV4101 interfaces to a PCI target that supports bursts, then the target should issue a stop at the end of a burst line, since the EV4101 address wraps and the PCI target does not. STOPN remains asserted until FRAMEN is HIGH.

TRDYN **Target Data Ready** **Bidirectional, im/xm=i/o**
When asserted LOW, TRDYN indicates to the initiator during writes that the target can receive data. During reads, a LOW on TRDYN indicates the target is driving valid data. Data is transferred when both IRDYN and TRDYN are asserted.

6.2 Miscellaneous Signals

ARBMODE2P **Arbitration Mode** **Input**
When ARBMODE2P is deasserted LOW, the EV4101 is in Arbitration Mode 1. In this mode, the bus arbiter is internal to the EV4101. When ARBMODE2P is asserted HIGH, the EV4101 is in Arbitration Mode 2, where the arbiter is external to the EV4101. This mode is used when the EV4101 is not the Central Processor, but is part of an add-on card within a host computer system.

BCLKP **Buffered Clock** **Output**
BCLKP is the buffered PURE_CLKP output. BCLKP either operates at the full PURE_CLKP frequency (HALFP tied LOW) or at the half PURE_CLKP frequency (HALFP tied HIGH). AC timing information is referenced to this clock.

BCLKRSTN **BCLKP Reset** **Input**
When BCLKRSTN is LOW, it synchronously resets BCLKP to LOW. When BCLKRSTN goes HIGH, the BCLKP starts clocking on the rising edge of PURE_CLKP. This is used to start BCLKP on either rising edge number 1 or 2, when BCLKP is running half the frequency of PURE_CLKP. This pin should be held HIGH during normal operation.

BIG_ENDIANP
Big/Little Endian Select **Input**
When BIG_ENDIANP is tied HIGH, the EV4101 interprets instructions and data in big endian format. When BIG_ENDIANP is tied LOW, the EV4101 interprets instructions and data in little endian format.

BRESETN **Synchronous System Reset** **Input**
Assertion of BRESETN resets the EV4101. BRESETN should be held asserted for a minimum of ten clock cycles. Reset puts all the configuration registers into a

known state. All other registers, such as the Cause Register, Status Register, the register file, and caches, are left in an unknown state.

CWAITIP **Wait for Interrupt** **Output**
This signal is driven HIGH when the Wait for Interrupt instruction (WAITI) has been executed. CWAITIP is driven LOW when an interrupt is encountered. The WAITI instruction reduces power in the EV4101 while it waits for an interrupt. Other devices can also opt to go into a power saving mode by monitoring CWAITIP.

HALFP **Clock Divider** **Input**
This input should be tied HIGH or LOW, and should not be changed dynamically. When HALFP is tied HIGH, the external interface operates at half the PURE_CLKP clock frequency. When HALFP is tied LOW, the external interface operates at the PURE_CLKP frequency.

ICECLKP **Rx and Tx Bit Rate Clock x 16** **Input**
This signal is the common receive and transmit clock. It runs at 16 times the bit-rate frequency of the received and transmitted bit stream, which is up to 16 MHz for the SerialICE application. ICECLKP is not required to run during reset.

ICERXP **Rx Received Bits** **Input**
This signal is the serial input received bit stream.

ICETXP **Tx Transmitted Bits** **Output**
This signal is the serial output transmitted bit stream. The output is changed on the positive edge of ICECLKP.

INT[5:0] **Condition/Interrupt** **Input**
INT[1:0] are interrupt pins. INT[5:2] are used as either condition pins or interrupt pins. The internal BBCC configuration register determines whether the inputs are condition or interrupt pins. INT[0] is combined with the DBERR bit in the SCR, and INT[2] is combined with the RxRDY bit in the Rx Status Register of the SerialICE port. For further description, please see Section 4.2, “SerialICE Port Overview,” and Section 4.4, “System Configuration Register.”

IP_DN **Instruction/Data Transaction** **Output**
The EV4101 drives IP_DN HIGH when the transaction is fetching an instruction, and drives IP_DN LOW when the EV4101 is reading or writing data. This output is valid only when the EV4101 is the initiator of the transaction.

PURE_CLKP **System Clock** **Input**
PURE_CLKP is the system clock. It has a 70/30 or 30/70 percent duty cycle.

T0_INTN **Sticky Bit** **Output**
The timer asserts this signal when Timer 0 is programmed as an interrupt generator; that is, it is in interrupt mode and Timer 0 reaches zero.

T0_OUTN **Timer 0** **Output**
T0_OUTN can be programmed so that when Timer 0 counts down to zero, T0_OUTN is either asserted LOW, or toggled from its previous value.

T1_OUTN **Timer 1** **Output**
T1_OUTN can be programmed so that when Timer 1 counts down to zero, T1_OUTN is either asserted LOW, or toggled from its previous value.

WBURSTN **Write Burst Priority** **Input**
After the EV4101 performs a write transaction, it checks if WBURSTN is LOW. If WBURSTN is LOW, then other write transactions queued in the write buffer are given highest priority over both Read-Instruction and Read-Data transactions previously queued, even if the Read-Priority bit in the BBCC Configuration Register is set.

6.3 Test Signals

The following signals are associated with silicon test pins that have no functional meaning to the user. However, some of them must be tied to a certain value for normal operation. For connections, see Figure 12, page 64.

CHSTSTP **Cache Test Enable** **Input**
When high, hardware cache test is enabled. This pin should be held LOW during normal operation.

CTEST_RFWEP

Register File Write Enable **Input**

When LOW, writes to the CPU register file are disabled. This is used when the debug tool is active. This pin should be held HIGH during normal operation.

IDDTN **IDD Test** **Input**

When LOW, all static power consuming devices are turned off. This pin should be held HIGH during normal operation.

PMON_OUTP **Process Monitor** **Output**

Output for process monitor cell. This pin should be left UNCONNECTED during normal operation.

SE **Scan Chain Loading Enable** **Input**

Assertion of this signal enables scan chain loading and unloading. This pin should be held LOW during normal operation.

SI **Scan Chain Input** **Input**

Inputs the scan chain. This pin should be left UNCONNECTED during normal operation.

SO **Scan Chain Output** **Output**

Outputs the scan chain. This pin should be left UNCONNECTED during normal operation.

TN **3-State Outputs** **Input**

Assertion of this input causes all of the EV4101 outputs and bidirectional signals to be 3-stated. This pin should be held HIGH for normal operation.

TST **Scan Test Enable** **Input**

When high, the SCAN test is enabled. This pin should be held low during normal operation.

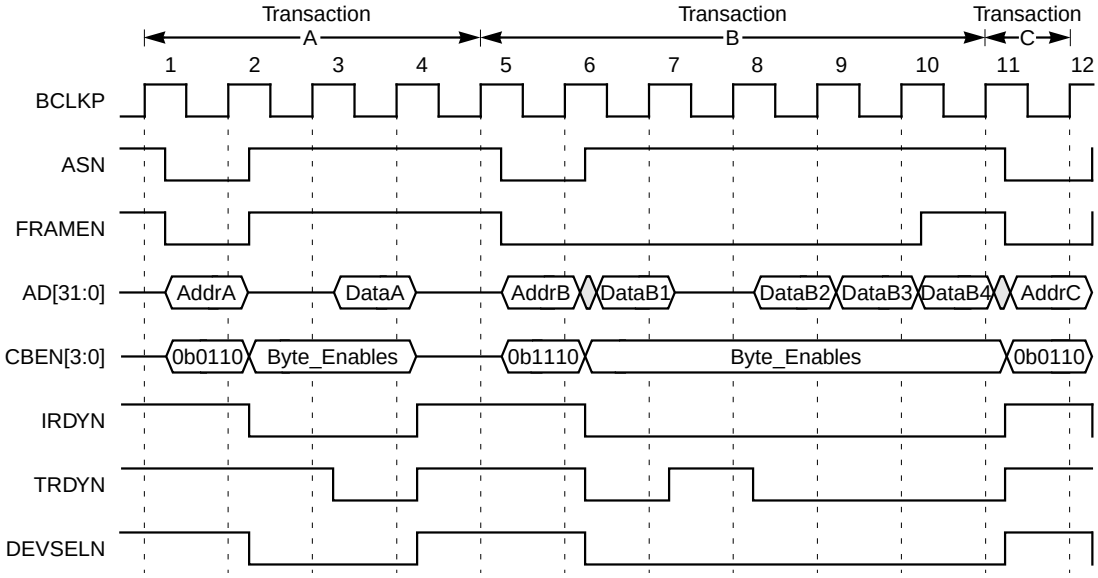
7 Bus Interface

The EV4101 chip bus protocol follows the basic PCI (Peripheral Component Interface) protocol, which allows seamless connection to components employing the PCI interface. The EV4101 bus interface operates at either the same frequency as the microprocessor or at half the frequency. The three transactions that the EV4101 chip performs when it is the bus initiator are Memory Read, Memory Write, and Memory Line Read.

7.1 Single Word Read and Burst Read Operations

Figure 4 shows a single word read followed by a four-word burst read. A description of each operation follows the figure.

Figure 4 Single Word Read Followed by Four-Word Burst Read



7.1.1 Transaction A: Single Word Read without Contention on the AD Bus

Clock Cycle 1 – The assertion of FRAMEN indicates that a new transaction is starting. The address is driven onto AD[31:0], and command information is driven onto CBEN[3:0]. (0b0110 indicates a

memory read transaction.) ASN is also asserted to indicate that the AD bus is driven with a valid address, although ASN is not part of the PCI protocol.

Clock Cycle 2 – The system should latch the address from AD and the transfer direction from CBEN0 on the rising edge of Clock Cycle 2. CBEN[3:0] is driven with the byte enables to indicate which ports are active for both reads and writes.

The Initiator asserts IRDYN to indicate that it can receive data on the next rising clock edge. Because this is a read operation, the target has opted to allow an idle cycle on the AD[31:0] by keeping TRDYN negated, as required by the PCI specification (though not mandatory for the EV4101). The target must drive DEVSELN within six clocks of FRAMEN to acknowledge that the target exists and will respond; otherwise, a bus error will result.

Clock Cycle 3 – The target may now add as many wait states as needed, before driving TRDYN and data. In this example, the target drives data so that it is valid by the rising edge of Clock Cycle 4, the fastest possible per the PCI specification.

Clock Cycle 4 – The assertion of IRDYN and TRDYN indicates that data is transferred on this rising clock edge. The deassertion of FRAMEN indicates that this is the final data transfer for this transaction.

An idle cycle occurs when both FRAMEN and IRDYN are negated, allowing a turnaround cycle for the AD bus. The PCI specification allows a one clock turnaround cycle to prevent contention on the AD bus. To override this idle cycle, set the FAST bit in the BBCC Configuration Register. (The FASTP bit is bit 24 of the SCR and defaults to zero for PCI compliancy.) The waveform shows that this bit is cleared, and an idle cycle allows the turnaround on the AD bus.

7.1.2 Transaction B: Four-Word Burst Read with Contention on the AD Bus

Clock Cycle 5 – Transaction B begins after the rising edge of Clock Cycle 5. The command phase, when 0b1110 is driven on CBEN[3:0], indicates a Memory Line Read.

Clock Cycle 6 – The target does not follow PCI protocol in this example. It drives data and TRDYN immediately after Clock Cycle 6. If the target

did not support bursts, it would have also asserted STOPN to indicate that the first data transfer would be the last data transfer of the transaction.

Clock Cycle 7 – The EV4101 latches the data on the rising edge of Clock Cycle 7. The target also deasserts TRDYN HIGH to indicate that a wait state is needed before Word 2 is transferred.

Clock Cycle 8 – The Initiator does not latch data, since TRDYN is sampled as negated. TRDYN and AD are driven off of Clock Cycle 8, so that the Initiator samples the data on Clock Cycle 9.

Clock Cycle 9 – Word 2 is latched. Since FRAMEN is sampled LOW, another transfer will occur.

Clock Cycle 10 – Word 3 is latched. Since FRAMEN is sampled LOW, another transfer will occur.

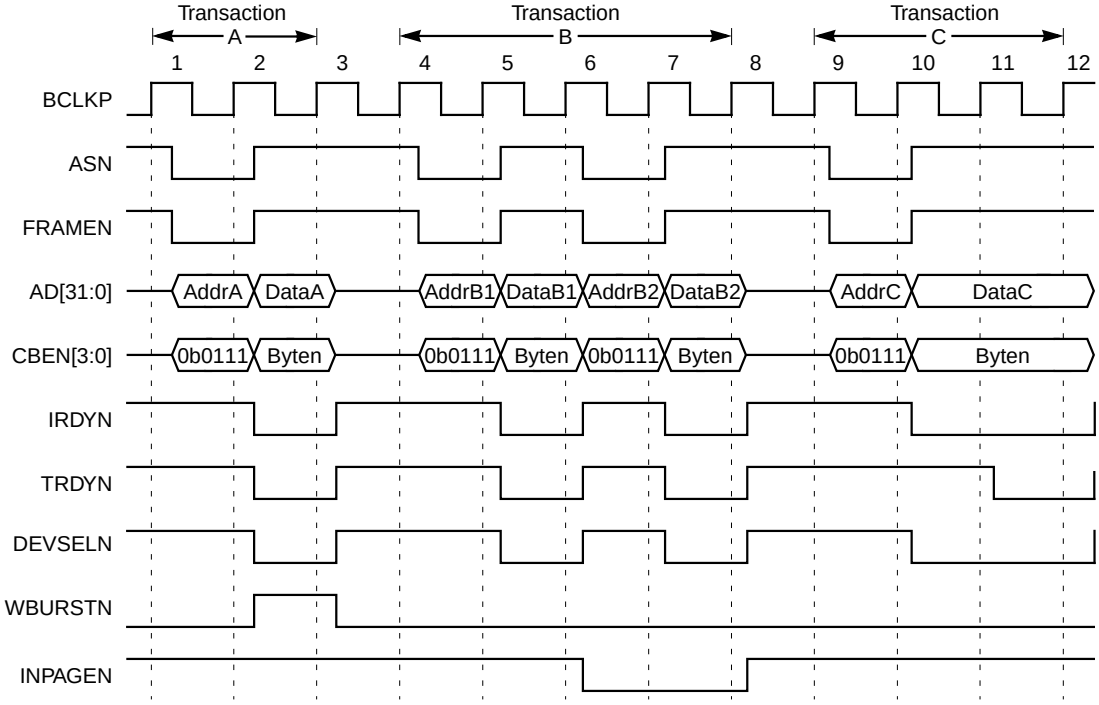
7.1.3 Transaction C: Single Word Read with FASTP Asserted

Clock Cycle 11 – This example shows what will happen if the FASTP bit of the SCR is set to one. Word 4 latched. Since FRAMEN is sampled HIGH, this is the last word of the burst. At Clock Cycle 11, another transaction begins. The complete transaction is not shown, since the only interesting part is the potential bus contention at the edge of Clock Cycle 11.

7.2 Single Word Writes and Burst Writes

Figure 5 shows Transaction A, a single word write, followed by Transaction B, first and second words of a 2-word burst write, and Transaction C, a single word write with wait state. A description of each operation follows the figure.

Figure 5 Single Word and Two-Word Burst Writes



7.2.1 Transaction A: Single Word Write

Clock Cycle 1 – A write transaction begins with the assertion of FRAMEN, AD[31:0], and CBEN[3:0]. CBEN[3:0] = 0b0111 indicates a write.

Clock Cycle 2 – Data is driven in Clock Cycle 2 and IRDYN is asserted, indicating that the write data is valid. The target drives TRDYN, indicating it is ready to accept data. The target also drives WBURSTN HIGH indicating that it does not want the EV4101 chip to give burst writes higher priority over reads.

Clock Cycle 3 – Because both IRDYN and TRDYN are LOW, data is transferred on the rising edge of Clock Cycle 3.

7.2.2 Transaction B1: First Word of a Two-Word Burst Write

Clock Cycle 3 – An idle cycle is generated on Clock Cycle 3, because the address of the transaction is not on the same page. (The BBCC Configuration Register determines the page size.) This idle cycle is needed, because the target is driving TRDYN and DEVSELN HIGH during Clock Cycle 3, as required by the PCI active deassertion protocol. The target 3-states these signals at the rising edge of Clock Cycle 4. If an idle cycle was not inserted, and the transaction was intended for a different target, then contention may have occurred just after the rising edge of Clock Cycle 4, as the old target 3-states its TRDYN, and another target begins to drive TRDYN.

Clock Cycle 4 – A write transaction begins. Note that writes are always single word writes, each with their own address phase.

Clock Cycle 5 – The target responds by asserting TRDYN. The target also drives WBURSTN LOW, indicating that if multiple, same page writes exist in the write buffer, then the EV4101 should give them priority and perform the back-to-back writes.

Clock Cycle 6 – Data B1 is transferred on the rising edge of Clock Cycle 6. WBURSTN is sampled on the rising clock edge where TRDYN is asserted.

7.2.3 Transaction B2: Second Word of a Two-Word Burst Write

Clock Cycle 6 – Transaction B2 begins without an idle cycle (defined as a cycle where both FRAMEN and IRDYN are HIGH). An idle cycle is not required when the second write is from the same initiator and to the same target. The EV4101 chip also calculates INPAGEN from the address to support RAS-parking targets.

The second burst write is not guaranteed when WBURSTN is asserted. It is merely a request that the EV4101 prioritize back-to-back writes over instruction or data fetches.

Clock Cycle 7 – Write data is driven, along with the appropriate byte enables.

Clock Cycle 8 – Data B2 is transferred on the rising edge of Clock Cycle 8.

7.2.4 Transaction C: Single Word Write with Wait State

Clock Cycle 8 – Idle cycle. (If WBURSTN were deasserted at the rising edge of Clock Cycle 8, and the next transaction was a write and on the same page, a fast back-to-back transaction would have been performed. As specified by the PCI specification, all targets must support fast back-to-back writes.)

Clock Cycle 9 – The write transaction begins.

Clock Cycle 10 – The target asserts DEVSELN, indicating that it will eventually respond. TRDYN is HIGH, creating a wait cycle.

Clock Cycle 11 – TRDYN is deasserted. Data is transferred on the rising edge of Clock Cycle 12.

---

8 Retry Support

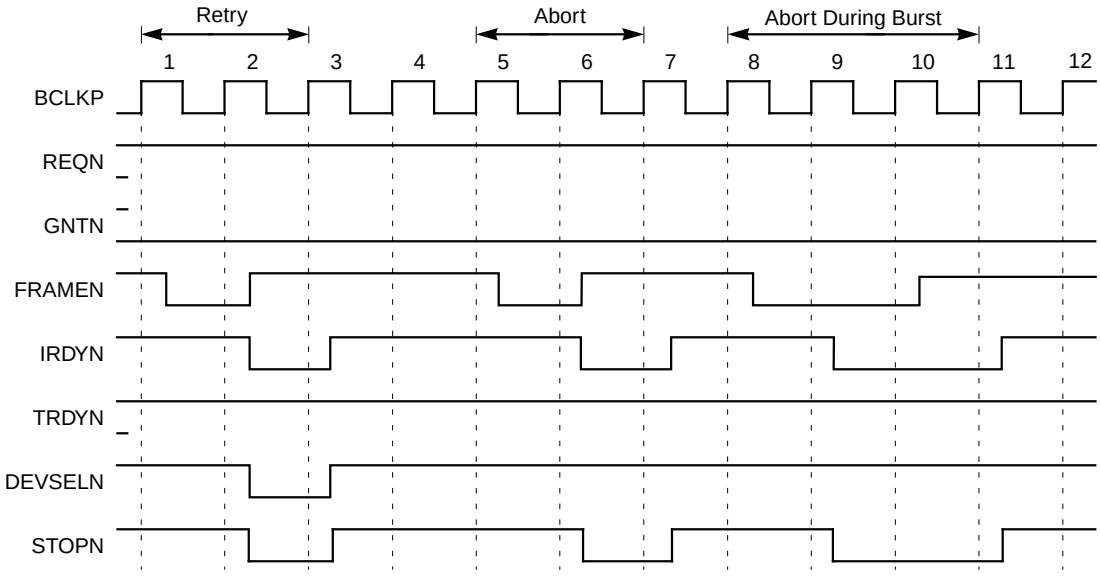
A PCI target issues a retry to a transaction that it cannot fulfill by asserting both STOPN and DEVSELN, while TRDYN remains deasserted. To respond to a retry, the EV4101 relinquishes ownership of the bus, allowing another PCI agent to become a master on the PCI bus and perform a transaction. The EV4101 then requests ownership of the bus, and either repeats the original transaction or performs a different transaction. There is no guarantee that the original transaction will be tried consecutively after a retry, because the EV4101 may have a different transaction that has a higher priority by the time it reacquires the bus.

If the retry is issued to the EV4101 during the middle of a Memory Line Read, then the EV4101 may opt not to retry the transaction. This situation occurs because a Memory Line Read is used to fill the cache with additional instructions/data that may not have been implicitly requested by EV4101. Note that the EV4101 option not to retry deviates from the PCI protocol, which states that retries must be retried.

The PCI specification also describes a situation where a retry may be issued in the middle of a burst transaction, when the target feels that it is taking too long to fulfill the transaction (called the “slow target disconnect”). This situation is *not* supported by the EV4101, and results in a bus error, if attempted.

Figure 6 illustrates a retry on a single word attempt, followed by the retry of the same transaction on Clock Cycle 5, only to receive an abort at Clock Cycle 7. The EV4101 starts a Memory Line Read on Clock Cycle 8, which is aborted on Clock Cycle 10. Note that STOPN must remain asserted until FRAMEN is sampled, deasserted, and IRDYN asserted.

Figure 6 Retry



9 Bus Error Support (Abort)

The EV4101 stops the bus transaction and issues a bus error when it receives the assertion of STOPN in conjunction with the deassertion of both DEVSELN and TRDYN. If a bus error occurs on a burst read, the remaining words are not read. The words that have already been transferred to the cache are marked as valid.

10 Burst Accesses

The EV4101 performs an address wraparound when performing a memory Line Read. If the cache refill size is programmed to be 8 words and the first access is to address 0x8, then the EV4101 requests the

subsequent words in the following order: 0xC, 0x10, 0x14, 0x18, 0x1C, 0x0, 0x4. This method is different from the method described in the PCI specification's Cacheline Wrap Mode, which describes the similar access to be in the following order: 0x8, 0xC, 0x0, 0x4, 0x18, 0x1C, 0x10, 0x14. Therefore, the Cacheline Wrap Mode, as described by the PCI specification, is not used. Instead, the burst order called Linear Incrementing is used, as indicated when AD[1:0] is 0x0 during the address phase.

A conflict results where the EV4101 is expecting data from a wrap, and the target is providing the data from a linear sequence. It is the responsibility of the target to stop the transaction at the final word before the disparity begins.

Note: If the block refill size is programmed to four, then when AD[3:2] = 0b11, STOPN should be asserted with the final TRDYN. If the block refill size is programmed to eight, when AD[4:2] = 0b111, STOPN should be asserted with the final TRDYN. Since AD is a multiplexed bus, it will not contain address information other than the starting word. It is the target's responsibility to increment its private address.

11 Arbitration

The EV4101 chip supports two different arbitration modes (Mode 1 and Mode 2). In Mode 1, the EV4101 is the arbiter and is used when it is the central microprocessor of a system. Mode 2 is used when the EV4101 is a microprocessor on an add-on card. In this mode, the microprocessor requests access to the bus before starting each transaction. The ARBMODE2P input determines which arbitration mode is in effect.

11.1 Mode 1: REQN (Input), GNTN (Output)

In this mode, the EV4101 is both the controller and master. The EV4101 can begin a transaction at any time it is not asserting GNTN. When an external master needs the bus, it drives REQN. When EV4101 loses ownership of the bus, it 3-states its bus signals (FRAMEN, IRDYN, CBEN, AD), and asserts GNTN to the new master.

The external master acknowledges ownership of the bus when FRAMEN is asserted, and relinquishes ownership when both FRAMEN and IRDYN are deasserted. This EV4101 protocol is 100% compatible with PCI arbitration protocol.

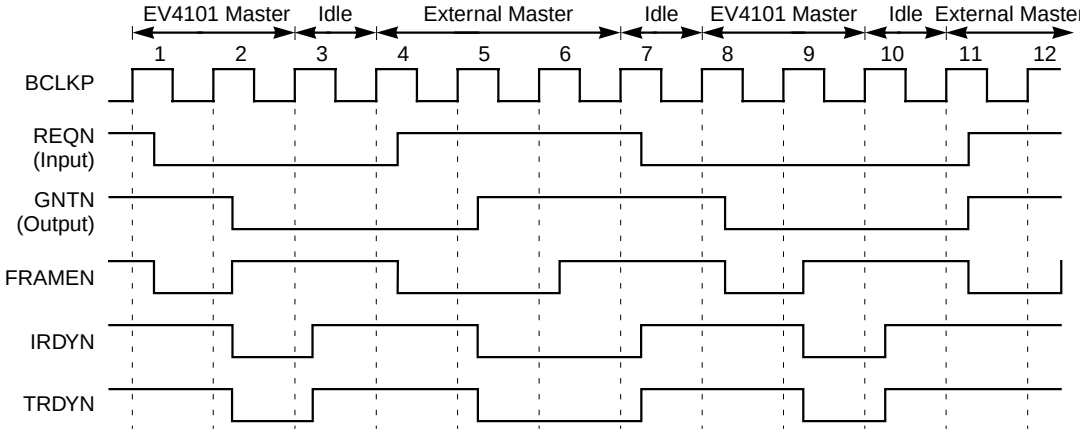
11.2 Mode 2: REQN (Output), GNTN (Input)

The arbiter is external to EV4101 in this mode. When EV4101 needs to perform a transaction on the external bus, it asserts REQN until GNTN is asserted. The EV4101 does not support atomic operations, such as READ, MODIFY, WRITE. The EV4101 owns the external bus from the moment that it asserts FRAMEN, until it completes its transaction.

Hidden arbitration is supported with this protocol, whereby the external arbiter may grant the bus to another master while the EV4101 is performing a transfer. Even though GNTN is asserted while the EV4101 is in the middle of a transaction, it still maintains control of the bus until its final TRDYN. (TRDYN asserted when FRAMEN is negated.) The second master must monitor FRAMEN and IRDYN, and may not begin its transaction until FRAMEN and IRDYN are both HIGH.

Figure 7 shows a Mode 1 arbitration example.

Figure 7 Bus Arbitration: Mode 1



Clock Cycle 1 – To start a transaction, the EV4101 asserts FRAMEN at Clock Cycle 1. At the same time, an external master asserts REQN to request ownership of the bus.

Clock Cycle 2 – The EV4101 asserts GNTN to the external device, while continuing with the EV4101-initiated data transfer.

Clock Cycle 3 – The external device samples GNTN as asserted, but must wait until it also samples both FRAMEN and IRDYN HIGH before it is allowed to start a transaction.

Clock Cycle 4 – The external device samples FRAMEN HIGH, IRDYN HIGH, and GNTN LOW, so it is allowed to start a transaction off of Clock Cycle 4.

Clock Cycles 5–6 – The External Master continues with its transaction.

Clock Cycle 7 – Final data is transferred, and the external master asserts REQN to request the bus again.

Clock Cycle 8 – The EV4101 asserts GNTN to the external device, and starts an EV4101-initiated transaction.

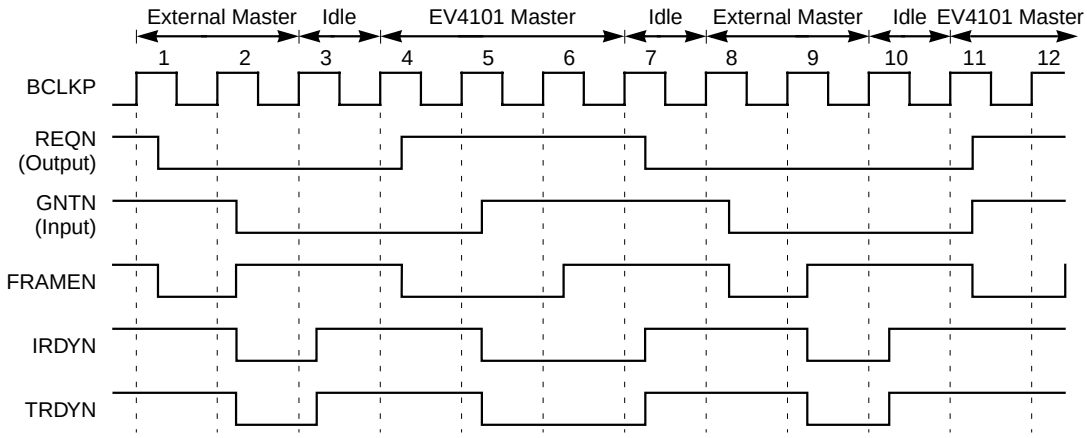
Clock Cycle 9 – The external device samples GNTN, but since both FRAMEN and IRDYN are not HIGH, it waits.

Clock Cycle 10 – The EV4101 continues with its transaction.

Clock Cycle 11 – Because the external device samples GNTN LOW and both FRAMEN and IRDYN HIGH, it starts its transaction. Note that the EV4101 may deassert GNTN at any time, but because the external device just sampled GNTN LOW at this clock edge, it is allowed to start its transaction. If GNTN had been deasserted a clock earlier, then the external device should not start its transaction.

Figure 8 is a Mode 2 arbitration example.

Figure 8 Bus Arbitration: Mode 2



Clock Cycle 1 – The EV4101 asserts REQN to acquire bus mastership. An external master asserts FRAMEN off of Clock Cycle 1 to start a transaction.

Clock Cycle 2 – The external arbiter asserts GNTN to the EV4101, while the external device continues with its data transfer.

Clock Cycle 3 – The EV4101 samples GNTN as asserted, but must wait until it also samples both FRAMEN and IRDYN HIGH before it is allowed to start a transaction.

Clock Cycle 4 – The EV4101 samples FRAMEN HIGH, IRDYN HIGH, and GNTN LOW. It is allowed to start a transaction off of Clock Cycle 4.

Clock Cycle 5–6 – The EV4101 continues its transaction.

Clock Cycle 7 – The final data is transferred, and the EV4101 asserts REQN to request the bus again.

Clock Cycle 8 – The external arbiter asserts GNTN to the EV4101. Simultaneously, an external master starts a transaction.

Clock Cycle 9 – The EV4101 samples GNTN, but because both FRAMEN and IRDYN are not HIGH, it waits.

Clock Cycle 10 – The external device continues its transaction.

Clock Cycle 11 – Because the EV4101 samples GNTN LOW and both FRAMEN and IRDYN HIGH, it starts its transaction. Note that the external arbiter may deassert GNTN at any time, but because the EV4101 just sampled GNTN LOW at this clock edge, it is allowed to start its transaction. If GNTN had been deasserted a clock earlier, then the EV4101 would not have started its transaction.

12 Inpage Support

The PGSZ (Page Size) bits in the EV4101 Configuration register allows external DRAM controllers to park their RAS signals. The EV4101 calculates and drives the INPAGEN signal whenever the current address is within the same page as the previous transaction address.

Note: INPAGEN is always deasserted on the first word after regaining control from another master.

13 Bus Snooping Support

Bus snooping is only supported in Host Mode. It is not supported in Add-in Card Mode, because the EV4101 must be the arbiter, in order to guarantee access to the caches when an external master is performing a transfer.

When an external device is performing a write to memory, the EV4101 snoops on the transaction. The EV4101 supports only one address latch, which is non-PCI compliant. It is implicitly assumed that DMA transfers are always to cacheable targets, such as memory.

In order for a DMA transaction to be snooped, the DMA burst must not be larger than four or eight words, depending on the EV4101 cache line configuration. If the starting address is not aligned on one of these boundaries, then the first burst must be less than four/eight words, because the snoop is only effective on one cache line at a time, and only one cache line per PCI address phase.

The target must monitor the status of the SDONEP signal, and not assert TRDYN until after it samples SDONEP HIGH. Because the EV4101 has a write-through cache, if the target has a Snoop Back-Off (SBO) input, it should tie SBO HIGH, indicating a clean snoop.

14 Specifications

This section specifies the EV4101 electrical characteristics and has two subsections:

- AC Timing
- Electrical Requirements

14.1 AC Timing

This section describes the AC timing characteristics of the EV4101's interface. The timing relationships between BCLKP and various EV4101 signals that constitute its interface are depicted in Figures 9 through 11. The timing is shown in Tables 27 through 29. All numbers are obtained through measurements on the EV4101.

Figure 9 Clock Timing (PURE_CLKP)

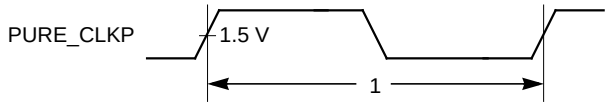


Figure 10 Input Timing

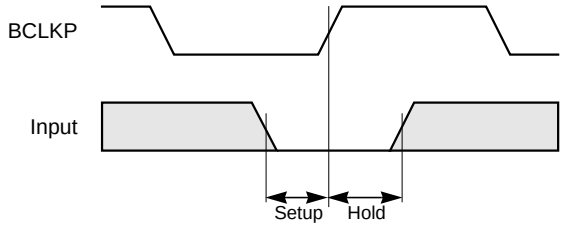


Figure 11 Output Timing

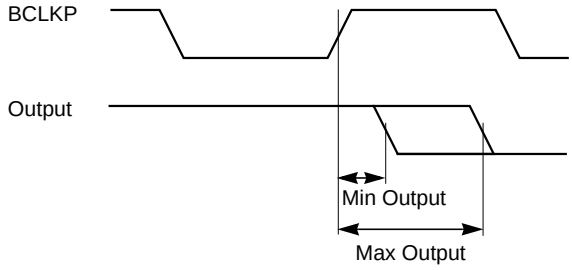


Table 27 EV4101 PURE_CLKP Input Timing

Parameter	Inputs	Min	Max	Units
1	PURE_CLKP Frequency (70/30 or 30/70 percent duty cycle)	–	66	MHz

Table 28 EV4101 AC Input Timing (Alphabetical)

Parameter	Inputs[1]	Setup	Hold	Units
2	AD[31:0]	6.7	4.9	ns
3	BRESETN	2.5	4.1	ns
4	CBEN[3:0]	20.0	4.4	ns
5	DEVSELN	4.1	3.5	ns
6	FRAMEN	20.0	4.3	ns
7	GNTN	2.4	3.2	ns
8	INT[5:0]	2.6	4.1	ns
9	IRDYN	20.0	4.0	ns
10	REQN	0.4	3.7	ns
11	STOPN	4.6	4.1	ns
12	TRDYN	4.2	4.3	ns
13	WBURSTN	3.7	3.1	ns

1. With respect to rising edge of BCLKP with 50 pF load.

Table 29 EV4101 AC Output Timing (Alphabetical)

Parameter	Outputs[1]	Min	Max	Units
14	AD[31:0]	2.1	20.2	ns
15	ASN	2.0	15.6	ns
16	CBEN[3:0]	1.8	15.5	ns
17	CWAITIP	1.8	9.8	ns
18	DEVSELN	1.1	8.4	ns
19	FRAMEN	1.7	15.5	ns
20	GNTN	1.6	12.3	ns
21	INPAGEN	1.9	15.8	ns
22	IP_DN	2.1	10.9	ns
23	IRDYN	1.3	8.7	ns
24	REQN	3.1	14.6	ns
25	SDONEP	2.8	12.6	ns
26	T0_INTN	2.1	12.3	ns
27	T0_OUTN	1.8	10.8	ns
28	T1_OUTN	1.9	10.2	ns
29	TRDYN	1.1	8.2	ns

1. With respect to rising edge of BCLKP with 50 pF load.

14.2 Electrical Requirements

This section specifies the electrical requirements for the EV4101. Four tables list electrical data in the following categories:

- Absolute Maximum Ratings (Table 30)
- Recommended Operating Conditions (Table 31)
- Capacitance (Table 32)
- DC Characteristics (Table 33)

Table 30 Absolute Maximum Ratings

Symbol	Parameter	Limits[1]	Unit
V_{DD}	DC Supply	–0.3 to +3.9	V
V_{IN}	Input Voltage	V_{DD} + 0.3	V
I_{IN}	DC Input Current	±10	mA
T_{STG}	Storage Temperature Range, Plastic	–40 to +125	°C

1. Referenced to V_{SS}.

Table 31 Recommended Operating Conditions

Symbol	Parameter	Limits	Unit
V_{DD}	DC Supply, Commercial	+3.15 to +3.45	V
T_A	Ambient Temperature	0 to +70	°C

Table 32 Capacitance

Symbol	Parameter[1]	Min	Typ	Max	Unit
C_{IN}	Input Capacitance	2.5	2.5	2.5	pF
C_{OUT}	Output Capacitance	2.0	2.0	2.0	pF
C_{IO}	I/O Bus Capacitance	2.5	2.5	2.5	pF

1. Measurement conditions are V_{IN} = 3.3 V, T_A = 25 °C, and clock frequency = 1 MHz.

Table 33 DC Characteristics

Symbol	Parameter	Condition[1,2]	Min	Typ	Max	Units
V_{IL}	Voltage Input Low			–	0.8	V
V_{IH}	Voltage Input High		2.0	–	–	V
V_{OH}	Voltage Output High	I_{OH} = –6.0 mA	2.5	3.0	V_{DD}	V
V_{OL}	Voltage Output Low	I_{OL} = 6.0 mA	0.0	0.2	0.5	V
I_{IL}	Current Input Leakage	V_{DD} = Max, V_{IN} = V_{DD} or V_{SS}	–10	±1	10	µA
I_{OZ}	Current 3-State Output Leakage	V_{DD} = Max, V_{OUT} = V_{SS} or V_{DD}	–10	±1	10	µA
I_{IPU}	Current Input Pull-up	V_{IN} = V_{SS}	–80	–145	–200	µA
I_{OZU}	Current 3-State Output w/Pull-up	V_{IN} = V_{SS}	–80	–145	–200	µA
I_{OSP}	Current P-Channel Output Short Circuit (6 mA Output Buffers)[2]	V_{DD} = Max, V_{OUT} = V_{SS}	–60	–40	–30	mA
I_{OSN}	Current N-Channel Output Short Circuit (6 mA Output Buffers)[2]	V_{DD} = Max, V_{OUT} = V_{DD}	25	40	70	mA
I_{DD}	Quiescent Supply Current	V_{IN} = V_{DD} or V_{SS}	< 1	1–2	60	µA
I_{CC}	Dynamic Supply Current	V_{DD} = Max, f = 66 MHz	–	150	–	mA

1. Specified V_{DD} at ambient temperature over the specified range.
2. Not more than one output may be shorted at a time for a maximum duration of one second.

15 Pinout, Package, and Ordering Information

The EV4101 is available in a 100-pin thin quad flat pack (TQFP) package. Table 34 provides ordering information for the EV4101. Table 35 provides a pin description summary and Table 36 provides an alphabetical pin list for the 100-pin TQFP.

Table 34 EV4101 Ordering Information

Part Number	Order Number	Clock Frequency (MHz)	Package Type	Operating Range
EV4101	L9C0099	66	100-pin TQFP	Commercial

Table 35 Alphabetical Pin Description Summary

Mnemonic	Description	Type[1]	Drive (mA)	Active
AD[31:0]	Address/Data Bus	3-State Bidirectional, C, PU, F	6	HIGH
ARBMODE2P	Arbitration Mode	Input, PU, F	–	HIGH
ASN	Address Strobe	3-State Output, F	6	LOW
BCLKP	Buffered Clock	3-State Output, F	6	HIGH
BCLKRSTN	BCLKP Reset	Input, PU, F	–	LOW
BIG_ENDIANP	Big/Little Endian Select	Input, PU, F	–	HIGH
BRESETN	Synchronous System Reset	Input, PU, F	–	LOW
CBEN[3:0]	Command/Byte Enable Bus	3-State Bidirectional, C, PU, F	6	LOW
CSHTSTP	Cache Test Enable	Input, PU, F	–	HIGH
CTEST_RFWEP	Register File Write Enable	Input, PU, F	–	HIGH
CWAITIP	Wait for Interrupt	3-State Output, F	6	HIGH
DEVSELN	Device Select	3-State Bidirectional, C, PU, F	6	LOW
FRAMEN	Frame	3-State Bidirectional, C, PU, F	6	LOW
GNTN	Bus Grant	3-State Bidirectional, C, PU, F	6	LOW
HALFP	Clock Divider	Input, PU, F	–	HIGH
ICECLKP	Rx and Tx Bit Rate Clock x16	Input, PU, F, Schmitt Trigger	–	Rising Edge
ICERXP	Rx Received Bits	Input, PU, F, Schmitt Trigger	–	HIGH
ICETXP	Tx Transmitted Bits	3-State Output, F, Maximum Slew Rate Control	4	HIGH

(Sheet 1 of 2)

Table 35 Alphabetical Pin Description Summary (Cont.)

Mnemonic	Description	Type[1]	Drive (mA)	Active
IDDTN	IDD Test	Input, PU, F	–	LOW
INPAGEN	Inpage Transaction	3-State Output, F	6	LOW
INT[5:0]	Condition/Interrupt	Input, PU, F	–	HIGH
IP_DN	Instruction/Data Transaction	3-State Output, F	6	HIGH
IRDYN	Initiator Data Ready	3-State Bidirectional, C, PU, F	6	LOW
PMON_OUTP	Process Monitor	Output	–	–
PURE_CLKP	System Clock	Input, PU, F	–	Rising Edge
REQN	Bus Request	3-State Bidirectional, C, PU, F	6	LOW
SDONEP	Snoop Done	3-State Output, F	6	HIGH
SE	Scan Chain Loading Enable	Input	–	HIGH
SI	Scan Chain Input	Input	–	HIGH
SO	Scan Chain Output	3-State Output, F	6	HIGH
STOPN	Stop Bus Transaction	Input, PU, F	–	LOW
T0_INTN	Sticky Bit	3-State Output, F	6	LOW
T0_OUTN	Timer 0	3-State Output, F	6	LOW
T1_OUTN	Timer 1	3-State Output, F	6	LOW
TN	3-State Outputs	Input, PU, F	–	LOW
TRDYN	Target Data Ready	3-State Bidirectional, C, PU, F	6	LOW
TST	Scan Test Enable	Input, PU, F	–	HIGH
WBURSTN	Write Burst Priority	Input, PU, F	–	LOW

(Sheet 2 of 2)

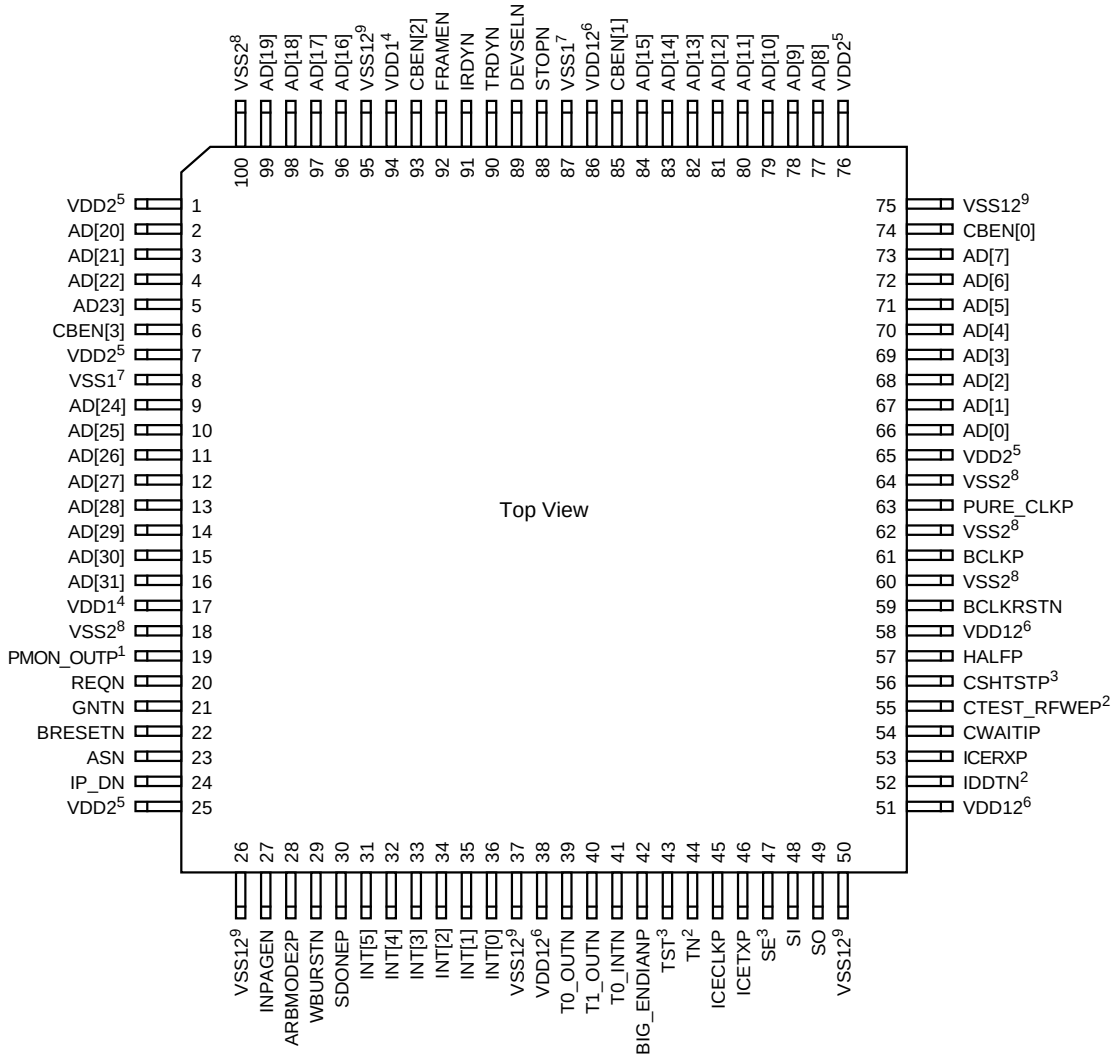
1. PU = internal pull-up resistor; F = 5-volt compatible; C = standard LVTTL voltage levels.

Table 36 Alphabetical Pin List for the 100-pin TQFP

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
AD[0]	66	AD[27]	12	ICERXP	53	TST[1]	43
AD[1]	67	AD[28]	13	ICETXP	46	VDD1[4]	17
AD[2]	68	AD[29]	14	IDDTN[2]	52	VDD1[4]	94
AD[3]	69	AD[30]	15	INPAGEN	27	VDD2[5]	1
AD[4]	70	AD[31]	16	INT[0]	36	VDD2[5]	7
AD[5]	71	ARBMODE2P	28	INT[1]	35	VDD2[5]	25
AD[6]	72	ASN	23	INT[2]	34	VDD2[5]	65
AD[7]	73	BCLKP	61	INT[3]	33	VDD2[5]	76
AD[8]	77	BCLKRSTN	59	INT[4]	32	VDD12[6]	38
AD[9]	78	BIG_ENDIANP	42	INT[5]	31	VDD12[6]	51
AD[10]	79	BRESETN	22	IP_DN	24	VDD12[6]	58
AD[11]	80	CBEN[0]	74	IRDYN	91	VDD12[6]	86
AD[12]	81	CBEN[1]	85	PMON_OUTP[3]	19	VSS1[7]	8
AD[13]	82	CBEN[2]	93	PURE_CLKP	63	VSS1[7]	87
AD[14]	83	CBEN[3]	6	REQN	20	VSS2[8]	18
AD[15]	84	CSHTSTP[1]	56	SDONEP	30	VSS2[8]	60
AD[16]	96	CTEST_RFWEP[2]	55	SE[1]	47	VSS2[8]	62
AD[17]	97	CWAITIP	54	SI	48	VSS2[8]	64
AD[18]	98	DEVSELN	89	SO	49	VSS2[8]	100
AD[19]	99	FRAMEN	92	STOPN	88	VSS12[9]	26
AD[20]	2	GNTN	21	T0_INTN	41	VSS12[9]	37
AD[21]	3	HALFP	57	T0_OUTN	39	VSS12[9]	50
AD[22]	4	ICECLKP	45	T1_OUTN	40	VSS12[9]	75
AD[23]	5			TN	44	VSS12[9]	95
AD[24]	9			TRDYN	90	WBURSTN	29
AD[25]	10						
AD[26]	11						

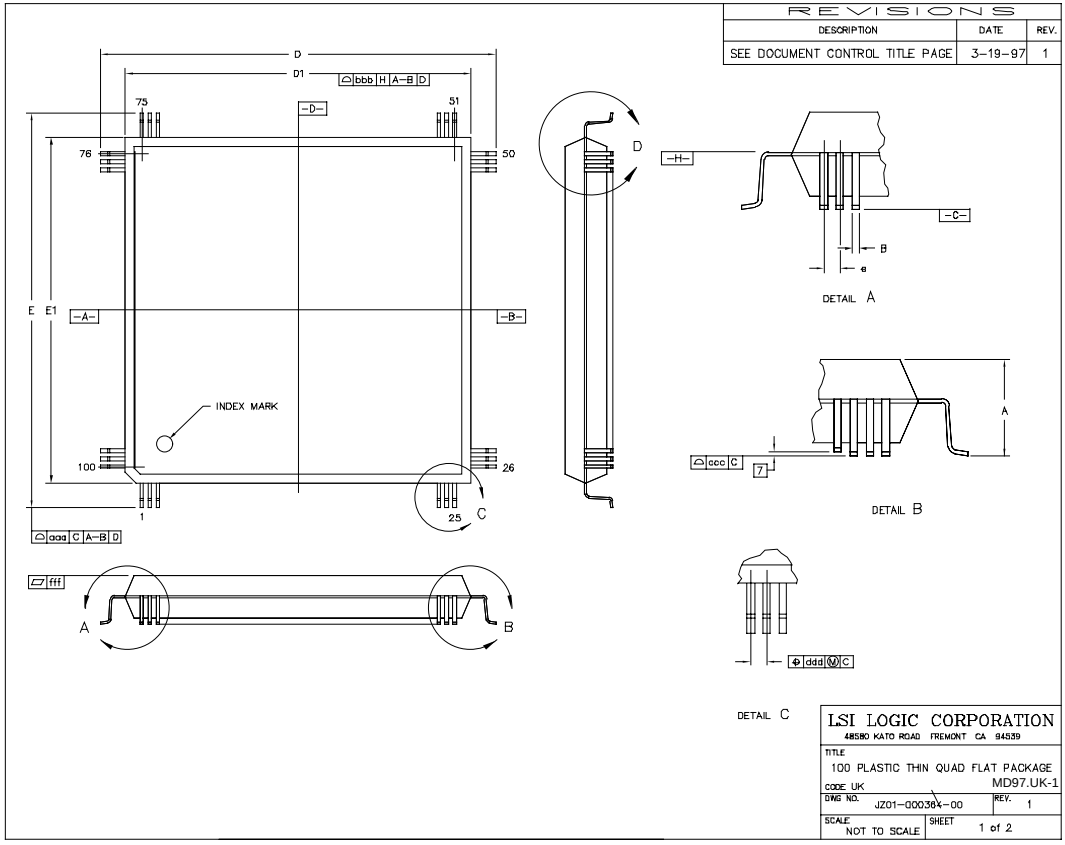
1. Connect this pin through a 4.7 kΩ resistor to VSS.
2. Connect this pin through a 4.7 kΩ resistor to VDD.
3. Used for factory test only. Leave unconnected.
4. VDD for core.
5. VDD for pad frame.
6. VDD for both core and pad frame.
7. VSS for core.
8. VSS for pad frame.
9. VSS for both core and pad frame.

Figure 12 100-Pin TQFP Pinout Diagram



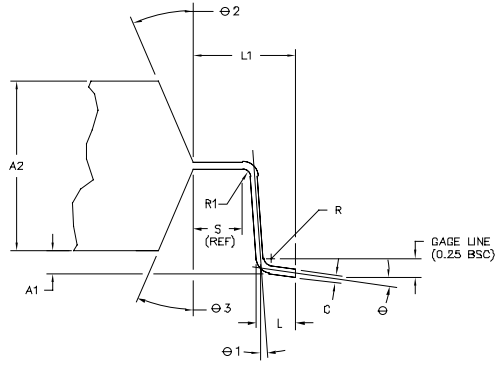
1. Used for factory test only. Leave unconnected.
2. Connect this pin through a 4.7 kΩ resistor to VDD.
3. Connect this pin through a 4.7 kΩ resistor to VSS.
4. VDD for core.
5. VDD for pad frame.
6. VDD for both core and pad frame.
7. VSS for core.
8. VSS for pad frame.
9. VSS for both core and pad frame.

Figure 13 100-Lead TQFP (UK) Mechanical Drawing



Important: **This drawing may not be the latest version. For board layout and manufacturing, obtain the most recent engineering drawings from your LSI Logic marketing representative by requesting the outline drawing for package code PE.**

Figure 13 (Cont.) 100-Lead TQFP (UK) Mechanical Drawing



DETAIL D

DIMENSIONS IN MM				
SYM	MIN	NOM	MAX	NOTE
A			1.60	
A1	0.05		0.15	
A2	1.35	1.40	1.45	
B	0.170	0.220	0.270	6, 10
B1	0.170	0.200	0.230	
C	0.090	0.150	0.200	10
D	15.90	16.00	16.10	2
D1	13.90	14.00	14.10	3
e		0.5 BSC		8
E	15.90	16.00	16.10	2
E1	13.90	14.00	14.10	3
L	0.45	0.60	0.75	
L1		1.00		
N		100		
R	0.08	0.15	0.20	
R1	0.08			
S	0.20			
fff			0.05	11
Q	0	3.5	7	
Q1	0			
Q2	11	12	13	
Q3	11	12	13	
aaa			0.20	
bbb			0.20	
ccc			0.076	
ddd			0.08	

NOTES: UNLESS OTHERWISE SPECIFIED

1 DATUM PLANE –H– LOCATED AT MOLD PARTING LINE AND IS COINCIDENT WITH THE BOTTOM OF LEAD, WHERE THE LEAD EXITS THE PLASTIC BODY

2 TO BE DETERMINED AT SEATING PLANE –C–.

3. DIMENSION D1 AND E1 DO NOT INCLUDE MOLD PROTUSION. ALLOWABLE PROTUSION IS .25mm PER SIDE.

4 THESE DIMENSIONS TO BE DETERMINED AT DATUM PLANE –H–.

5 ALL DIMENSIONS IN MILIMETERS. CONTROLLING DIMENSION IN MILIMETER. INCHES SHALL BE ROUNDED TO NEAREST .001 INCH.

6 DIMENSION B DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.05mm TOTAL IN EXCESS OF THE B DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT MINIMUM SPACE BETWEEN ADJACENT LEADS TO BE 0.07mm.

7 COPLANARITY OF ALL LEADS SHALL BE WITHIN 0.076 mm DIFFERENCE BETWEEN HIGHEST AND LOWEST LEAD WITH SEATING PLANE –C– AS REFERENCE.

8 LEAD PITCH DETERIMINED AT DATUM –H–.

9 MARKING AREA FREE OF PROTRUSION AND INTRUSION

10 PLATING THICKNESS INCLUDED. PLATING THICKNESS TO BE 0.005 mm MIN; 0.020 mm MAX.

11 FROM TRUE LEAD LOCATION MEASURED AT SEATING PLANE –C–.

12 DATUM –A–, –B– AND –D– DETERMINE CENTERLINE OF LEADFRAME

LSI LOGIC CORPORATION
48580 KATO ROAD FREMONT CA 94539

TITLE: 100 PLASTIC THIN QUAD FLAT PACKAGE
CODE UK MD97.UK-2
DWG NO. JZ01-000364-00 REV. 1
SCALE NOT TO SCALE SHEET 2 of 2

Important: **This drawing may not be the latest version. For board layout and manufacturing, obtain the most recent engineering drawings from your LSI Logic marketing representative by requesting the outline drawing for package code PE.**

Notes

Notes

Headquarters

LSI Logic Corporation
North American Headquarters
Milpitas CA
Tel: 408.433.8000
Fax: 408.433.8989

LSI Logic Europe Ltd.
European Headquarters
Bracknell, England
Tel: 44.1344.426544
Fax: 44.1344.481039

LSI Logic K.K.
Headquarters
Tokyo Japan
Tel: 81.3.5463.7821
Fax: 81.3.5463.7820

Visit us at our web site: http://www.lsilogic.com

ISO 9000 Certified

Printed on Recycled Paper

Printed in USA
Order No. C15021.B
Doc. No. DB09-000029-02