Integrated Device Technology, Inc.
2975 Stender Way, Santa Clara, CA - 95054
Phone #: (408) 727-6116 Fax #: (408) 727-2328

Errata Notification

EN #: IEN01-02 Errata Revision #: 11/5/01

Issue Date: December 14, 2001 Effective Date: November 5, 2001

Product Affected: IDT79RC32T355

ERRATA DETAILS: **ATTACHMENT** ☒ Yes ☐ No

- ☐ Data Sheet Errata
- ☒ Device Errata
- ☐ Device Manual Errata
- ☐ Application Notes Errata
- ☐ Other (Please Specify)

IDT Web Site Location:

http://www.idt.com/products/pages/Integrated_Processors-79RC32355.html

CUSTOMER NOTIFICATION:

As a result of Errata listed above, IDT has decided to notify you through this Errata Notification.
We recommend you to review this notification in details prior to the use of affected product in your application.
Please use acknowledgement below or E-Mail to request additional information.

If you have any questions, please feel free to contact your local IDT distributor or sales representative.

Customer: _____ E-Mail Address: _____

Name/Date: _____ Phone #: _____

Title: _____ Fax # : _____

CUSTOMER COMMENTS: _____

Notes

Supplemental Information

This Device Errata is for the RC32355 device and supplements information found in the IDT79RC32355 User Reference Manual.

Revision History

October 9, 2001: Initial Publication, items #1 through #15.

October 30, 2001: Added explanation of System Revision Register in "System Identification Register for ZC Silicon" section.

November 5, 2001: Expanded explanation of item #1.

System Identification and Revision Registers for ZC Silicon

To determine the vendor, implementation, and revision of an integrated communications processor, use the System Identification and Revision Registers. For a complete description of these registers, see Chapter 1, "RC32355 Device Overview" in the RC32355 User Reference Manual.

The System Revision Register has been updated in ZC silicon to read the following:

Revision = 0x2 (ZC)

The System Identification Register reads the following:

Implementation = 0x1 (RC32355)

Vendor = 0x0 (Integrated Device Technology)

Device Errata for Revision ZC Silicon

Description and Workaround

Item #1 - Padding of Undersized Packets by the Ethernet MAC Causes a Transmit Failure (Reference # 046)

Issue: The Ethernet transmit locks up if a collision occurs to the padded portion of an undersized packet. Only undersized packets (less than 64-bytes) are padded, legal-sized packets (greater than or equal to 64-bytes) are transmitted as is. The problem does not occur with legal-sized packets, since they are not padded.

The Ethernet interface consists of an Ethernet MAC core and some control logic between the MAC core and the CPU and DMA. The control logic handles all data transfers to and from the MAC. The MAC handles the 802.3 bus protocols, such as collision detection and data padding. The control logic passes the packet data to the MAC while watching for a collision detection from the MAC.

If a packet is undersized, the control logic sends the few bytes to the MAC while watching for a collision detection from the MAC. The MAC will then add bytes to the undersized packet (pad) until it equals 64-bytes. The problem happens because the control logic only watches for a collision while the few bytes are sent out, not while the undersized packet is being padded. So the collision detection is missed if the collision happens during the padded portion of this packet.

Workaround: There are two ways to solve this problem: eliminate undersized packets or eliminate collisions.

In full duplex mode, transmit and receive packets occur over separate wires, so collisions are eliminated. Consequently, this problem does not occur in full duplex mode.

In half duplex mode, transmit and receive packets happen on the same wire, so collisions could occur and create this problem. Since the problem happens only when undersized packets are passed to the MAC and are padded there, the problem can be solved if undersized packets are eliminated. To eliminate undersized packets, software must ensure that all packets are at least 64-bytes (i.e., software pads undersized packets instead of the MAC). Undersized packets are thus eliminated and this problem will not happen in half duplex mode.

Fix: There are no plans to fix this item.

Item #2 - Ethernet TX Core Lockup During Jam (Reference # 065)

Issue: If Ethernet transmit asserts the JAM bit (in ETHINTFC register) twice in a row without transmitting a packet between these two assertions, it may potentially lock up the Ethernet TX engine.

Workaround: Do not set JAM to one.

Fix: There are no plans to fix this item.

Item #3 - USB Erroneously Requests Data (Reference # 070)

Issue: The USB peripheral module allows more than 2 packets through USB TX when ITS control is set to one in register USBEPGC. When ITS is set, after the first packet is transferred from memory to USB FIFO (through DMA), the DMA waits for the devcs (from USB peripheral) before moving to packet number 2. In this case, the USB peripheral sends a “devcs_ignore” to the DMA, and the DMA can queue packets in the FIFO. However, even after two packets are present in the FIFO (the max allowed by the RC32355), the USB peripheral will continue to request additional packets.

Workaround: The RC32355 operates correctly if ITS is cleared to zero (allowing only one packet to be queued in the FIFO).

Fix: There are no plans to fix this item.

Item #4 - Improper Setting/Clearing of FD Bit for Ethernet Receive Descriptors (Reference # 077)

Issue: The FD bit in the DEVCS register of the first descriptor of a packet in the descriptor list is not set. However, the FD bit does get set in descriptors that are not the first descriptor.

Workaround: The FD bit should not be used to find the first descriptor for an Ethernet packet. The FD bit will be set in descriptors that are neither the first nor last descriptor for the packet and may not be set in the first descriptor. Instead, use the finished (F) and done (D) bits in the first word of the descriptor. Finished will be set in a buffer that has been used if it is not the last one; done will be set in the last descriptor for the packet.

Fix: There are no plans to fix this item.

Item #5 - Ethernet Transmits Wrong Size Packet (Reference # 069)

Issue: Ethernet transmits the wrong size packet when the ITS control bit is set in the ETHINTFC register. Setting the ITS bit allows data from more than one packet to be written to the transmit FIFO. When the ITS bit is set, data from two packets in the FIFO is mingled causing a long length packet. The problem does not occur if one packet is set up and transmitted at a time.

Workaround: Do not set the ITS bit in the ETHINTFC register.

Fix: There are no plans to fix this item.

Item #6 - ATM AAL5 PDUs are Occasionally Received with CRC32 Errors Over DMA Channel 0 (Reference # 102)

Issue: First, to give some background information, the RC32355 can receive ATM cells over its ATM interface and transfer them to the local memory using one of nine DMA channels. CRC is computed by the hardware. The selection between CRC32 and CRC10 is indicated in the DEVCMD field of the DMA descriptor. If a connection is set up to receive AAL5 PDU over DMA channel 0, the CRC selection should be indicated for every cell being received. Typically, if an AAL5 PDU is being received, the DEVCMD in the current descriptor is set to one of the following ATM modes:

CRC32-first descriptor	First cell of the PDU
CRC32	Intermediate cell in the PDU
CRC32-last descriptor	Last cell of the PDU
CRC32-first and last descriptor	First and last cell of the PDU

If the DEVCMD is set to CRC32-first descriptor or CRC32-first and last descriptor, the RC32355 initializes the CRC32VAL register and the DEVCS field in the descriptor to 0xFFFF_FFFF. The DEVCS field holds the intermediate CRC32 value after the transfer of a cell. If the DEVCMD is set to CRC32 or CRC32-last descriptor, the RC32355 copies the intermediate CRC32 value from the DEVCS field to the CRC32VAL register before computing the CRC for the cell being received.

The problem occurs when AAL5 PDUs are received over DMA channel 0 and the DEVCMD fields in the DMA descriptors are set to either CRC32-first descriptor or CRC32-first and last descriptor. In this case, the PDUs are occasionally received with CRC32 errors.

Workaround: In all cases, set the DMA descriptor to receive the first cell of the PDU using:

DEVCS = 0xFFFF_FFFF

DEVCMD = CRC32 or CRC32-last descriptor ATM modes

This will cause the RC32355 to update the CRC32VAL register from the DEVCS field of the descriptor for every cell. The hardware will then automatically write the correct DEVCMD when the CRC is written to the DEVCS.

Fix: There are no plans to fix this item.

Item #7 - UART Receive FIFO Timeout Interrupt (Reference # 169)

Issue: The UART receive FIFO timeout interrupt does not work as stated. Specifically, the UART receive FIFO timeout interrupts are not masked/unmasked with the UART[0|1]IE register, field RDA (bit [0]).

Workaround: The UART receive FIFO time-outs are incorrectly masked/unmasked with UART[0|1]IE register, field RLS (bit [2]) instead of RDA (bit [0]). So, to enable the receive timeout interrupt in FIFO mode, UART[0|1]IE bit [2] should be set to one. A FIFO timeout interrupt will then occur if at least one character is in the receive FIFO and the most recent character was received more than 4 character times ago. This prevents bytes from sitting in the FIFO for long periods of time before the FIFO trigger level is reached.

Fix: There are no plans to fix this item.

Item #8 - UART Line Status Register: THR Bit (Reference # 170)

Issue: When the UART is in the 16550 (buffered/FIFO) mode, the user can expect register UART[0|1]LS, field THR, to be set to 1 when the transmit channel is ready to accept a new character for transmission. The 16550 specifications state that THR is guaranteed to be 1 when the Transmit Holding Register is empty and to be 0 when the Transmit Holding Register is not empty. However, this does not occur in the RC32355.

Workaround: Do not use THR by itself. Instead, use the THR and TE (Transmitter Empty) fields together. Use the following logic:

TE = 0, THR = 0
This means that Transmit Shift Register and Transmit Holding Register both have bytes in them (if in FIFO mode, at least one byte is in the FIFO).

TE = 0, THR = 1
Do not use this combination. It does not work.

TE = 1, THR = 0
Not valid.

TE = 1, THR = 1
Transmit Shift Register and Transmit Holding Register are both empty. OK to send next byte of data.

Fix: There are no plans to fix this item.

Item #9 - ICE Debug Boot Mode (Reference #174)

Issue: Debug Boot Mode control is part of the Boot Configuration Vector that is read into the RC32355 during cold reset. When this control is set high, the RISCore 32300 begins executing from address 0xFF20_0200 rather than 0xBFC0_0000 following a reset exception. This enables the CPU to boot from the ICE probe through the JTAG interface pins without using external memory. This feature has not been fully tested or validated.

Workaround: Accessing the Debug Boot Mode through the Boot Configuration Vector should not be used. Rather, the same functionality can be accomplished by forcing a debug exception in the EJTAG Control Register at reset time.

Fix: There are no plans to fix this errata.

Item #10 - USB Endpoint A Sometimes Stalls (Reference #173)

Issue: USB endpoint A occasionally gets stalled while trying to stall another endpoint. For example, when the host issues a SET_FEATURE command and tries to stall endpoint C, sometimes both endpoints A and C are stalled.

Workaround: Configuration from the host should be repeated until successful.

Fix: There are no plans to fix this item.

Item #11 - Ethernet Runt Packets of 1-63 Bytes in Length Not Discarded Properly (References #148, 175, 178, 179)

Issue: The Ethernet MAC on the RC32355 incorrectly handles the reception of some short Ethernet packets (packets that are less than 64 bytes in lengths) instead of discarding them. The behavior of the Ethernet MAC while receiving the sort packets depends also on the size of the Ethernet packets. There are two different cases:

A. Packets that are 1 to 60 bytes long.
Sometimes the Ethernet MAC does not properly discard packets that are 1 to 60 bytes long. Packets that are 1 to 5 bytes long are stretched to 6 bytes, which affects the byte count statistics, before being handled by the MAC.

Also, if there are multiple packets in the receive FIFO and one of them is less than 60 bytes, the byte count in the DEVCS of the packet before and after the short packet could be corrupted.

B. Packets that are 61 to 63 bytes long.
The Ethernet MAC on the RC32355 passes these packets through to the memory as if they were legal size packets. All the corresponding DMA pointers and statistics are updated accordingly. These packets do not generate any other internal side effects.

Workaround: The workaround consists of using software filters to either discard the packet or to recover from the error conditions generated by cases A and B. The three major filters include these operations:

- Check receive packet length to deal with the short packets
- Check for Erroneous Finished Flag to deal with the condition under which the receive DMA terminated
- Recover from the FIFO overrun error to deal with the overrun condition.

Receive Packet Length Check:

Description: The Ethernet driver performs this check while handling the receive interrupt (whenever a packet is available in the input FIFO). The length shown by the DEVCS field (referred to as "DEVCS Length") and that computed from the COUNT field in the control word of the DMA descriptor (referred to as "Buffer Length") may differ due to the reception and handling of short packets. To bypass this condition, the software handler for the receive interrupt first checks the length shown by the DEVCS field. If it is less than 64 bytes, the Ethernet packet is dropped and the involved buffers and descriptors are reclaimed. If the length shown by the DEVCS field is greater than 64 bytes but the length computed using the COUNT field of the control word and DEVCS length are different, then the Ethernet packet is also dropped.

Operation to be performed by the software filter:

1. If DEVCS Length < 64: Drop the Ethernet Packet.
2. If DEVCS Length >= 64 AND DEVCS != Buffer Length: Drop the packet.

In C Language:

```
if ( DEVCS_count < 64 )
passed_Flag = FALSE ; /* set the flag false if the packet length < 64, this would lead to dropping of this packet */
if ( DEVCS_count >= 64 && DEVCS_count != buffer_count)
passed_Flag = FALSE
```

In Pseudo code:

The software needs to compute the Buffer Length using the Control Word of the DMA descriptor.

Pointwise Pseudo Code:

- to compute the Buffer_count

    1. Read Control Word (dmaDescControl) of the DMA descriptor during receive in interrupt handler

    2. Buffer_count = (dmaDescControl & 0x1FFF - RX_DMA_COUNT) where RX_DMA_COUNT is the original count value loaded in the Control Word prior to DMA operation (say a value of 0x2000 which is greater than the maximum packet payload of 1514 decimal)

- 1st check:

    3. If (DEVCS_count < 64)

        Passed_Flag = FALSE

- 2nd check:

    4. IF (DEVCS_count >= 64 && DEVCS_count != Buffer_count)

        Passed_Flag = FALSE

This adds about 90 nanoseconds for every received packet processing, assuming the code to be residing in the cache, which would be true for most of the cases, and a 150MHz pipeline frequency.

Check for Erroneous Finished Flag:

Description: The Ethernet interface may start receiving packets without an end of packet DONE condition. Instead the MAC sets the Finished Flag bit.

The Ethernet Interface receives data which is DMA'ed to the receive buffers. On completion of each packet transfer, a DONE flag is set. If the receive buffer size is kept greater than the maximum Ethernet MTU size, then the Finished Flag would never be set in the normal course of receiving data over Ethernet. On the other hand, if the receive buffer size is less than the Ethernet MTU, at least two or more DMA descriptors and receive buffers would be needed for complete transfer of the Ethernet data. In this case, Ethernet Interface would encounter one or more valid Finished interrupt events based on the size of the receive buffer and the Ethernet payload.

The software fix for the above mentioned erroneous condition would be quite simple for the case where the receive buffer size is more than the Ethernet MTU size. The driver should incorporate checking of the FINISHED bit by enabling Interrupt On Finished (IOF) event in the DMA descriptor. On finding an illegal FINISHED event, the driver should reset/restart the Ethernet Interface. Because of the low overhead as part of the software fix, its recommended to use receive buffer size greater than the maximum legal Ethernet MTU size.

The software fix would be slightly more involved in the case of the receive buffer size being less than the Ethernet MTU size. Software has to keep a check on the number of FINISHED events seen per each received packet. A counter can be incremented when a FINISHED interrupt occurs and cleared to zero when a DONE interrupt occurs. If the FINISHED count exceeds the maximum packet size divided by the buffer size, the Ethernet interface should be reset.

Operation:

In case of receive buffer sizes in excess of Ethernet MTU size, check for the FINISHED flag bit in the Receive Interrupt Handler for every received interrupt (which would occur on DONE or FINISHED event). In C Language:

```
if ( IS_DMA_FINISHED(status) ) /* Status is
already read as part of the Receive Handler */
{
bnS355Restart(pDrvCtrl);
return;
}
```

FIFO Overrun Error:

Description: The FIFO Overrun condition occurs when the input FIFO overflows during the reception of mixed short and legal size packets. On FIFO Overrun condition, Ethernet MAC needs to be reset/restarted.

Operation: Incorporate interrupt handler for Ethernet Input Overflow. On receiving an interrupt due to an OVERRUN condition, reset the Ethernet MAC.

Fix: There are no plans to fix this item.

Item #12 - Support for DPI Mode Has Been Removed (Reference # 186)

Issue: The ATM SAR on the RC32355 can be configured to support four different types of physical interfaces: Utopia level 1, Utopia level 2, a single DPI, and a dual DPI. The single DPI and the dual DPI modes have not been validated on the device and will no longer be supported as part of the features set for the RC32355. Further, any reference to these two modes will be removed from the documentation.

Workaround: The ATM SAR needs to be configured as either Utopia level 1 or Utopia level 2.

Fix: There are no plans to fix this errata.

Item #13 - USB Stalls When Enabling or Disabling Endpoint Operating Mode (Reference # 154)

Issue: Some endpoint control bits programmed into the USBEP[B|C|D|E|F|G]C registers must pass between USB and system clock domains. Field MODE in these registers contains two binary bits where 0x0 is disabled, 0x1 is stall, 0x2 is busy, and 0x3 is enabled. Each MODE bit could arrive in the USB clock domain at slightly different times causing an unwanted stall condition (MODE equals 0x1) when going from enable to disable or disable to enable.

Workaround: Software should go to the busy MODE (0x2) first before enabling or disabling an endpoint. For example, always transition from enable to busy to disable or disable to busy to enable when changing modes. This will create a “one-hot” encoding of MODE that avoids the unintended stall mode.

Fix: There are no plans to fix this item.

Item #14 - Problem Enabling a Single PHY in Utopia II Mode (Reference # 185)

Issue: There is a problem in the ATM interface when only a single PHY is enabled in Utopia II mode. In that case, the Utopia II RX block in the RC32355 activates the PHY correctly, but the block incorrectly polls the PHY one more time before it suspends polling. A U2 compliant PHY is usually going to respond back with RXCLAV asserted (the problem doesn't exist if the PHY responds with RXCLAV deasserted). The ATM interface latches that assertion, and at the end of the current cell, it immediately activates the same PHY again. The PHY may or may not have a cell ready. If it does not, then it may hang or throw the polling sequence out of order.

Workaround: The fix for this is to always enable at least two PHYs when in Utopia II mode and put a pulldown on RXCLAV if the second PHY does not exist.

Fix: There are no plans to fix this item.

Item #15 - Software Cannot Detect an Illegal I2C Slave Start or Stop Condition (Reference # 192)

Issue: The I2C Slave interface should set an error indication if a start or stop condition is in an illegal position during an I2C bus transaction in which the slave is addressed. When an error is detected by the internal I2C Slave hardware, this error bit (ERR bit in the I2C Slave status register, I2CSS) should get set and remain set until cleared by software. However, the hardware implementation erroneously clears this bit one cycle after being set. This error condition can not reliably be detected by software and will not generate an interrupt.

The ERR bit in the I2C Master status (I2CMS) register, however, works properly.

Workaround: Do not use the ERR bit in the I2C Slave status register, I2CSS, or the interrupt that would be generated from it in the Interrupt Pending Register, IPEND5.

Fix: There are no plans to fix this item.

Issues Under Investigation

There are other items, not listed in this document, currently under investigation and not yet confirmed as errata. For a current list of such items, contact your local sales representative.