Ethernet MAC Device Driver Application Note V1.0

Publication Release Date: Aug. 2008

Support Chips:

NUC740A

NUC710A

NUC745A

Support Platforms:

All

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Table of Contents

1. Introduction

Nuvoton's Ethernet MAC Controller (EMC) consists of IEEE 802.3/Ethernet protocol engine with internal CAM function for Ethernet MAC address recognition; Transmit-FIFO, Receive-FIFO, TX/RX state machine controller and status controller. EMCs of NUC740A support both MII and RMII mode. The EMC of NUC710A/NUC745A only supports RMII (Reduced MII) interface to connect with PHY operating on 50MHz REF_CLK.
This application note describes some problems software engineers might have while developing EMC driver from scratch

2. Programming Notes

2.1. MISTA Status Handling

Both Tx and Rx descriptors and register MISTA would report error/abnormal status during EMC operating. Driver should read the status bits and make a proper action. Below list some status and the action should take, rest of status not listed here could be ignored during operation.

Table 2-1: MISTA Report Status

Status	Action
Runt Packet	Drop the packet. ARP in register should be cleared during normal operation, so this error should not happened
Packet Too Long	Drop the packet. ALP in register should be cleared during normal operation, so this error should not happened
CRC Error	Drop the packet. AEP in register should be cleared during normal operation, so this error should not happened
Tx Bus Error	Check the driver. This error occurs if Tx descriptors are corrupted. This should not happen during operation.
Tx Descriptor Unavailable	Do nothing. EMC just finished sending all packets in Tx descriptors.
Tx FIFO Underflow	If this error occurs often, modify TxTHD of register FFTCR to a higher level.
Rx Bus Error	Check the driver. This error occurs if Rx descriptors are corrupted. This should not happen during operation.
Rx Descriptor Unavailable (RDU)	Receive all packets and write any value to register RSDR. EMC run out of Rx descriptors and couldn't receive incoming packets. It is likely interrupt is block for too long, so check the whole system carefully.
Rx FIFO Overflow	If this error occurs often, modify RxTHD of register FFTCR to a higher level.

2.2. Frequently Seen Errors

Here list some errors possibly made by programmers and the consequences

Table 2-2: Errors and Causes

Symptom	Cause
Cannot access PHY	Forget to enable EnMDC bit of register MCMDR Forget to configure GPIO Set the wrong PHY address (The PHY address in driver must be consistent with hardware setting)
RDU status occurs	Block Rx interrupt for a long period
Bus error interrupt occurs	Mangle Rx/Tx descriptors

2.3. Cache

Nuvoton's ARM7 MCUs has cache controller built in. Applications can get better performance while cache is enabled. However, not everything could locate in cacheable memory space. Rx/Tx descriptors receive buffers and transmit buffer pointed by Rx/Tx descriptors respectively are some data should stay in non-cacheable area since these data structures will be accessed by DMA. Programs could access the non-cacheable address by setting the most significant bit to 1. Below is a sample code for initializing Rx/Tx descriptors.

```
#define NON_CACHE_FLAG 0x80000000

__align(16) volatile RXBD rx_desc[RX_DESC_SIZE];
__align(4)  volatile char rx_buf[RX_DESC_SIZE][PACKET_BUFFER_SIZE];

__align(16) volatile TXBD tx_desc[TX_DESC_SIZE];
__align(4)  volatile char tx_buf[TX_DESC_SIZE][PACKET_BUFFER_SIZE];

// Init Rx descriptors
```

```
 for ( i =0 ; i < RX_DESC_SIZE ; i++) {
       rx_desc[i].SL=RXfOwnership_DMA;
       rx_desc[i].buffer=(unsigned long)&rx_buf[i]|NON_CACHE_FLAG;
       rx_desc[i].next=(unsigned long)&rx_desc[i+1]|NON_CACHE_FLAG;
}
rx_desc[i-1].next=(unsigned long)&rx_desc[0]|NON_CACHE_FLAG;
priv.rx_ptr = (unsigned long)&rx_desc[0]|NON_CACHE_FLAG;
writeReg(REG_RXDLSA, (unsigned long)&rx_desc[0]|NON_CACHE_FLAG);

// Init Tx descriptors
for ( i = 0 ; i < TX_DESC_SIZE ; i++ ) {
       tx_desc[i].SL=0;
       tx_desc[i].mode=0;
       tx_desc[i].buffer=(unsigned long)&tx_buf[i]|NON_CACHE_FLAG;
       tx_desc[i].next=(unsigned long)&tx_desc[i+1]|NON_CACHE_FLAG;
}
tx_desc[i-1].next=(unsigned long)&tx_desc[0]|NON_CACHE_FLAG;
priv.tx_ptr = (unsigned long)&tx_desc[0]|NON_CACHE_FLAG;
writeReg(REG_TXDLSA, (unsigned long)&tx_desc[0]|NON_CACHE_FLAG)
```

3. Revision History

Version	Date	Description
V1.0	Aug. 2008	● Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur. Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.