APPLICATION NOTE

## 1. INTRODUCTION

The purpose of this application note is to describe the implementation of a PCI bus master 100 Base-TX Fast Ethernet node using MXIC' highly integrated single chip Fast Ethernet NIC controller MX98715. In details, this document presents product overview, programming guide, hardware design and layout recommendations that can help you to quickly and smoothly implement a Fast Ethernet adapter card.

As you can find in the MX98715 driver diskette, MXIC already provided a complete set of high quality drivers for easier and more efficient way to interface with MX98715 on the most popular Network Operating Systems. Nevertheless, there are still some special applications or environment not covered in the MX98715 driver diskette. Driver developers, however, could still refer to the section of driver programming guide to accomplish the required driver. It is recommended that you are familiar with the MX98715 data sheet before reading this guide.

## 2. PRODUCT OVERVIEW

The MX98715 implements the 10/100Mbps MAC layer and Physical layer on a single chip in accordance with the IEEE 802.3 standard.

The MX98715 highly integrates with direct PCI bus interface, including PCI bus master with DMA channel capability, direct EEPROM as well as Boot ROM interface, and large on chip transmit/receive FIFOs. Also, the MX98715 is equipped with intelligent IEEE802.3u-compliant Nway auto-negotiation capability allowing a single RJ-45 connector to link with the other IEEE802.3u-compliant device without re-configuration. To optimize operating bandwidth, network data integrity and throughput, the proprietary Adaptive Network Throughput Control (ANTC) function is implemented. For detailed product specification information, please refer to the MX98715 data sheet.

## 3. HARDWARE DESIGN CONSIDERATIONS

### 3.1 SYSTEM APPLICATION BLOCK DIAGRAM

A system block diagram for the MX98715 based Fast Ethernet adapter card is shown as following:

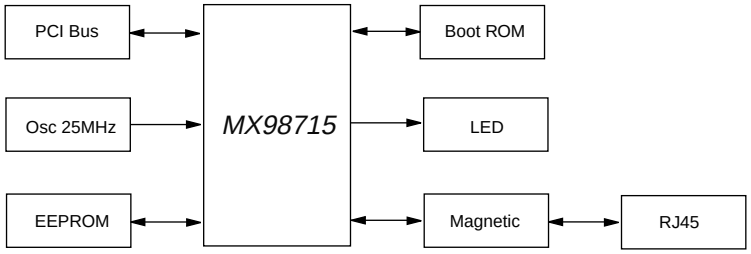



Fig. 1

### 3.2 PCI CONNECTION

The MX98715 provides direct PCI bus interface to PCI connector. Board designers should especially take care of the four pins of TDI,TDO,PRSNT1# & PRSNT2# that are only related to PCI bus connector. Boards that do not implement JTAG Boundary Scan should tight TDI and TDO together to prevent the scan chain from been broken.

Both pins PRSNT1# and PRSNT2# should be connected to ground indicating that the board is physically presenting in a PCI slot and providing information about the total power requirements ( less than 7.5W ) of the board.

### 3.3 OSCILLATOR

The MX98715 is designed to operate with a 25MHz oscillator module. The clock specification of this oscillator should meet 25MHz +/- 50PPM.

### 3.4 BOOT ROM

The MX98715 support a direct boot ROM interface allowing diskless workstations to remotely download operating system from network server. For proper operation, the access time of adapt EPROM should not excess 240ns.

### 3.5 SERIAL EEPROM

The MX98715 provides pins EECS,BPA0 (EECK), BPA1 (EEDI) and BPD0 (EEDO) for directly accessing the serial EEPROM. BPA0-1 and BPD0 serve as SK (EECK), DI (EEDI) and DO (EEDO) respectively. The contents of the EEPROM includes the ID information of the MX98715 (VendorID, DeviceID, Sub-vendorID, Sub-deviceID and MAC ID), and the configuration parameters for software driver. The EEPROM contents should be programmed according to MXIC's definition as mentioned in Appendix A. Detailed software programming example is described in section 4.5.

### 3.6 PROGRAMMABLE LED SUPPORT

The MX98715 provides two pins LED0 and LED1 to control display LEDs. Displayed messages are programmable through setting CSR9 bit-28 & bit-29 to serve as Activity /Linkspeed and Goodlink/Linkactivity LED respectively. The maximum sinking current of these output pins is 16mA. Current limiting resistor (560Ω) should be added to ensure proper operation. The following indicates the configuration setting table for LED display programming.

| CSR9 <28:29> | LED0 | LED1 |
|---|---|---|
| 00 | Activity | Goodlink |
| 11 | Link speed | Link Activity |

### 3.7 NETWORK INTERFACE TO MAGNETIC COMPONENT

For isolating and impedance matching purpose, an isolating transformer with 1:1 transmit and 1:1 receive turns ratio is required for transmit and receive twisted pair interface. In Appendix B, several transformers that we had verified successfully with MX98715 are listed for quick reference purpose.

### 3.8 OPTIMIZED EQUALIZER COMPONENTS

MXIC' Fast Ethernet solution utilizes adaptive equalizer to compensate the attenuation and phase distortion induced by different lengths of cable. To optimize transmit and receive signal quality, pins RTX and RTX2EQ should be connected to external resistors 560Ω (±1%)and 1.4KΩ (±1%) and then to ground respectively.

## 4. DRIVER PROGRAMMING GUIDE

This chapter will provide you the necessary information for programming driver for the MX98715 based node. Initialization module is introduced first that describes how MX98715 is initialized before any other operations can commence, then followed by actual implementation examples for both transmit and receive operations. Programming differences between MX98713,MX98713A and MX98715 are also included that will help you to upgrade your own driver to support all MXIC' NIC product series.

### 4.1 INITIALIZATION

```
  initializeTheTransmitRing()
  {
        unsigned int     i,j;
        unsigned long    physicaladdress;
        for (i=0; i<NumTXBuffers; i++) {
         /* memory allocation for tx descriptor_buffer (allign 4) */
         tx_resource[i]=
           (struct TX_RESOURCE *)((((unsigned int)tx_temp[i])+4)&
0xfffc);
        }

        for (i=0; i<NumTXBuffers; i++) {
         /* initialize the own bit to host tdes0 */
         tx_resource[i]->ownership=0x00;
         tx_resource[i]->tstatus=0x0000;
         tx_resource[i]->tdes0_unused=0x00;

         /* fill buffer_1_address tdes2 */
         get_ea((void far *)(tx_resource[i]->tx_buffer_data),
              &physicaladdress);
         tx_resource[i]->buff_1_addr=physicaladdress;

         /* fill buffer_2_address tdes3 */
         if (i==NumTXBuffers-1) j=0;
         else j=i+1;
         get_ea((void far *)(tx_resource[j], &physicaladdress);
         tx_resource[i]->buff_2_addr=physicaladdress;
        }
  }

  initializeTheReceiveRing()
{
        unsigned int     i,j;
        unsigned long    physicaladdress;
        for (i=0; i<NumRXBuffers; i++) {
         /* memory allocation for rx descriptor_buffer (allign 4) */
         rx_resource[i]=
           (struct RX_RESOURCE *)((((unsigned int)rx_temp[i])+4)&
0xfffc);
        }

        for (i=0; i<NumRXBuffers; i++) {
         /* set the own bit to chip  rdes0 */
         rx_resource[i]->frame_length=RDES0_OWN_BIT;
         rx_resource[i]->rstatus=0x0000;

         /* fill rdes1 */
                  rx_resource[i]->command=RDES1_BUFF-
RX_BUFFER_SIZE+rxpkt_size[i];

         /* fill buffer_1_address rdes2 */
         get_ea((void far *)(rx_resource[i]->rx_buffer_data),
              &physicaladdress);
         rx_resource[i]->buff_1_addr=physicaladdress;
         /* fill buffer_2_address rdes3 */
         if (i==NumRXBuffers-1) j=0;
         else j=i+1;
         get_ea((void far *)(rx_resource[j], &physicaladdress);
         rx_resource[i]->buff_2_addr=physicaladdress;
        }
  }

  initialize()
  {
        unsigned long  physicaladdress;

        NIC_read_reg(&csr6);
        NIC_write_reg(&csr6,csr6.value&(~(CSR6_SR|CSR6_ST)));
        delay(10);
        InitializeTheTransmitRing (6);
        InitializeTheReceiveRing (6);
        NIC_write_reg(&csr0,CSR0_L_SWR);
        delay(50);

        NIC_write_reg(&csr0,csr0shadow);
        get_ea((void far *)rx_resource[0],&physicaladdress);
        NIC_write_reg(&csr3,physicaladdress);
        get_ea((void far *)tx_resource[0],&physicaladdress);
```

```
        NIC_write_reg(&csr4,physicaladdress);
        NIC_read_reg(&csr16);
        NIC_write_reg(&csr7,csr7shadow);
        NIC_write_reg(&csr16,csr16shadow);
        //Clear status register
        NIC_write_reg(&csr5,(unsigned long)0xffffffff);
        NIC_write_reg(&csr6,csr6shadow);
        NIC_read_reg(&csr6);
        setup_frame(TDES1_SETUP_LAST,perfect);
  }
```

## 4.2 TRANSMITION MODULE

```
  bmtx()
  {
        unsigned char editmode, j;
        struct TX_RESOURCE *tx_pointer;

        initialize();
        fill_pattern(6);                        //fill pattern
        NIC_write_reg(&csr6,csr6.value&(~CSR6_ST));  //stop
        NIC_read_reg(&csr6);
          NIC_write_reg(&csr6,csr6.value|CSR6_SF);      //store and
forware
        NIC_read_reg(&csr0)
        NIC_write_reg(&csr0,csr0.value|0x020000);    //TAP=01
        tx_pointer=tx_resource[0];
        j=0;
        editmode=1;

        while (editmode) {
            if ((tx_pointer->ownership & 0x80)==0) {
              j++;
              j%=tx_pkt_num;
              if (tx_pointer->command & TDES1_LS_BIT)
               tx_error_detect(tx_pointer->tstatus);
              tx_pointer->ownership |= 0x80;
              tx_pointer=tx_resource[j];
            }
            if (kbhit()) {
              keycode_get();
              if (M_code!=0) {
                    switch (M_code) {
                       case 0x1b:       // ESC: quit
                             editmode=0;
                             break;
                       case 0x20:
                             NIC_read_reg(&csr6);
                          NIC_write_reg(&csr6,csr6.value^CSR6_ST);
                             break
                       default: break;
                    }
              }
            }
        }
  }
```

## 4.3 RECEPTION MODULE

```
  bmrx()
  {
        unsigned char editmode,i,j;
        unsigned long physicaladdress;
        struct RX_RESOURCE *rcv_pointer;

        initialize();
        rcv_pointer=rx_resource[0];
        j=0;
        editmode=1;

        while (editmode) {
            // if data received
            if ((rcv_pointer->frame_length & 0x8000)==0) {
              j++;
              j%=6;
              if (rcv_pointer->rstatus & RDES0_LS)
               rx_error_detect(rcv_pointer->rstatus);
              rcv_pointer->frame_length |= 0x8000;
              rcv_pointer=rx_resource[j];
            }
            if (kbhit()) {
              keycode_get();
              if (M_code!=0) {
                    switch (M_code) {
                       case 0x1b:       // ESC: quit
                             editmode=0;
                             break;
                       default: break;
                    }
              }
            }
        }
  }
```

### 4.4 CODING DIFFERENCE BETWEEN MX98713, MX98713A, MX98715 AND MX98725

#### 4.4.1 SPEED SELECTION

Speed selection for MX98713 is controlled by internal Nway registers. All the MII management commands should have the following structure:

<PRE><ST><OP><PHYAD><REGAD><TA><DATA><IDLE>

For detailed programming example, please refer to MX98713 application note.

As for MX98713A, MX98715 and MX98725, Internal NWay registers are removed and protocol selection is controlled by Operation Mode Register (CSR6) and 10Base-T Control Register (CSR14)

| | NWay Active | 100F | 100H | 10F | 10H |
|---|---|---|---|---|---|
| CSR6_PS | 0 | 1 | 1 | 0 | 0 |
| CSR6_PCS | X | 1 | 1 | X | X |
| CSR6_FD | 1 | 1 | 0 | 1 | 0 |
| CSR14_ANE | 1 | 0 | 0 | 0 | 0 |

#### 4.4.2 ELSE REGISTERS SETTING FOR DEVELOPING YOUR OWN DRIVER

There is an obvious change in setting the contents of CSR16 in comparison with MX98713. In MX98713, the offset 80h in IO space that is CSR16 must be set to “0x0f37XXXX” to bring this controller into normal operation mode before any initialization process can be started by driver. However, in MX98713A, MX98715 and MX98725, driver developers must set the CSR16 to be “0x0b3cXXXX” .

In summary, the contents of CSR16 for MXIC 100Base NIC controllers should be set differently as follow:

MX98713 = 0x0f37XXXX
MX98713A = 0x0b3cXXXX
MX98715 = 0x0b3cXXXX
MX98725 = 0x0b3cXXXX

Meanwhile, you could directly access the Nway auto-negotiation status from CSR20. Detailed format information please refer to MX98715 data sheet.

### 4.5 EEPROM ACCESSING

The following is a reference code for accessing the contents of EEPROM that stores ID information and node configuration for the MX98715.

```
/*************************************
*  Read all content from EEPROM
*************************************/
eeprom_read()
{
    unsigned int i, address, eeval;
    char bit;
    for (address=0; address<64; address++{
        NIC_write_reg(&csr9,(unsigned long)0x04800);
        eeprom_serial_in(0);
        eeprom_serial_in(1);      //command
        eeprom_serial_in(1);
        eeprom_serial_in(0);
        for(i=0; i<6; i++){       //address serial in
          bit = ((address>>(5-i)) & 0x01) ? 1:0;
          eeprom_serial_in(bit);
        }
        eeval=0;
        for(i=0; i<16; i++){      //dat serial out
          NIC_write_reg(&csr9,(unsigned long)0x04803);
          NIC_read_reg(&csr9);
         eeval += (((unsigned long)0x008 & csr9.value)>>3)<<(15-
i);
          NIC_write_reg(&csr9,(unsigned long)0x04801);
        }
        NIC_write_reg(&csr9,(unsigned long)0x04800);
        c46[address*2] = eeval & 0x0ff;
        c46[address*2+1] = (eeval >>8) & 0x0ff;
    }
}

/*************************************
*  Write a word to EEPROM
*************************************/
eeprom_write(unsigned int address, unsigned int data)
{
    unsigned int i;
    char bit;
    eeprom_wen();
    NIC_write_reg(&csr9,(unsigned long)0x04800);
    eeprom_serial_in(0);
    eeprom_serial_in(1);      //command
    eeprom_serial_in(0);
    eeprom_serial_in(1);
```

```
    for(i=0; i<6; i++){        //address serial in
      bit = ((address>>(5-i)) & 0x01) ? 1:0;
      eeprom_serial_in(bit);
    }
    for(i=0; i<16; i++){       //data serial in
      bit = ((data>>(15-i)) & 0x01) ? 1:0;
      eeprom_serial_in(bit);
    }
    NIC_write_reg(&csr9,(unsigned long)0x04800);
    NIC_write_reg(&csr9,(unsigned long)0x04801);
    i=0;
    do{
      i++;
      NIC_read_reg(&csr9);
    } while ((!(csr9.value & 0x08)) && (i<10000));
    NIC_write_reg(&csr9,(unsigned long)0x04800);
    if (i==10000) prstring ("Writing EEPROM error !!");
    eeprom_wds();
}

eeprom_wen()
{
    NIC_write_reg(&csr9,(unsigned long)0x04800);
    eeprom_serial_in(0);
    eeprom_serial_in(1);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(1);
    eeprom_serial_in(1);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    NIC_write_reg(&csr9,(unsigned long)0x04800);
}

eeprom_wds()
{
    NIC_write_reg(&csr9,(unsigned long)0x04800);
    eeprom_serial_in(0);
    eeprom_serial_in(1);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    eeprom_serial_in(0);
    NIC_write_reg(&csr9,(unsigned long)0x04800);
}

/*************************************
*  Serial inject a bit to EEPROM
*************************************/
eeprom_serial_in(unsigned int bit2)
{
    NIC_write_reg(&csr9,(unsigned long)0x04800+4*bit2);
    NIC_write_reg(&csr9,(unsigned long)0x04803+4*bit2);
    NIC_write_reg(&csr9,(unsigned long)0x04801+4*bit2);
}
```

## 5. PCB LAYOUT RECOMMENDATIONS

The MX98715-based adapter board is strongly recommended to separate the power plane into 3 regions, i.e., digital, analog and receive region to isolate from digital noise and noise coupling between the transmitter and the receiver. These power pins in these three regions are shown in the table listed below. Each VDD pin also needs a 0.1uF capacitor located as close to the VDD pin as possible.

| Analog Region | Receive Region | Digital Region |
|---|---|---|
| 81. VDD | 90. VDD | Others |
| 82. GND | 89. GND | |
| 84. VDD | 93. VDD | |
| 86. GND | 94. GND | |
| 88. VDD | 95. GND | |
| 87. GND | | |
| 96. VDD | | |
| 99. GND | | |
| 103. VDD | | |
| 104. GND | | |
| 105. GND | | |
| 106. VDD | | |
| 107. GND | | |
| 108. VDD | | |
| 109. GND | | |

All components of external oscillator circuits should be located very close to the CKREF pin and kept short traces length to avoid unwanted capacitance and inductance.

The power plane of designed PCB should be split into three regions. The below figure clearly describes an ideal power plane partition related to the MX98715's pinout. To reduce noise incurred among power traces of these three regions, designers should allocate Ferrit Beads on the interface of +5V traces between digital and analog regions as well as between analog and receive regions. Such placement of Ferrit Beads implies that the power traces between digital and receive regions should be clearly isolated; all the power for receive region should be from analog region. All the traces and resistance/capacitance components for these pins of MX98715 should be located in their specified regions. As for placement of the 25MHz oscillator, it is recommended to be apart from the receive and analog regions as far as possible to prevent from the impact of harmonic noise.

The MX98715 should be placed as close to the transformer as possible to reduce the length of layout traces and potential noise from coupling on RXIP/N and TXOP/N. The 100W resistor between RXIP and RXIN is suggested to be located as close to the RXIP and RXIN pins of MX98715 as possible. The 49.9W pull up (VDD) resistors for TXOP and TXON pins should be placed as close to the TXOP and TXON pins of MX98715 as possible for impedance match purpose. The receive paths (traces of RXIP/N) need to be in parallel and have equal routing length on the component side of the PCB, and absolutely do not have any through hole on the receive paths to keep the signals clearance.

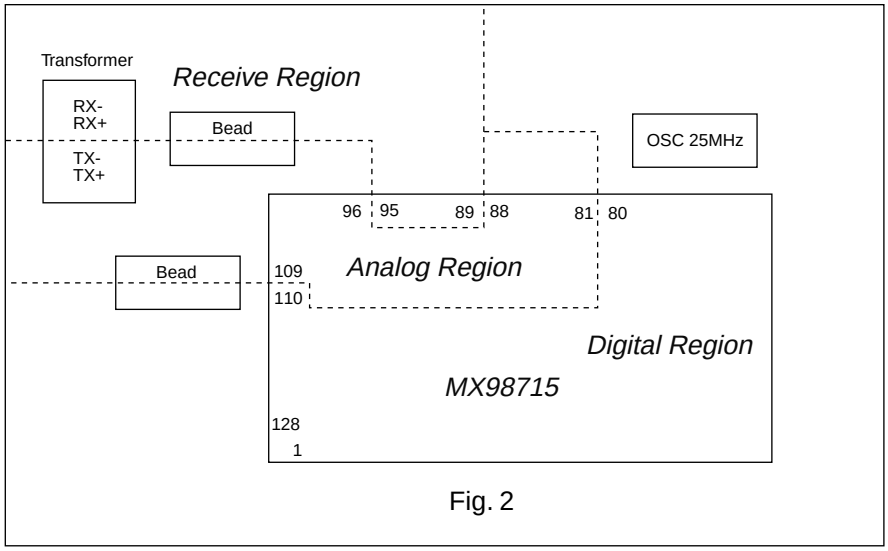



Fig. 2

## APPENDIX A: EEPROM FORMAT

| BYTE OFFSET (HEX) | DESCRIPTIONS |
|---|---|
| 00-19 | Reserved |
| 1a | Magic Packet ID Byte0 |
| 1b | Magic Packet ID Byte1 |
| 1c | Magic Packet ID Byte2 |
| 1d | Magic Packet ID Byte3 |
| 1e-39 | Reserved |
| 3a | Magic Packet ID Byte4 |
| 3b | Magic Packet ID Byte5 |
| 3c-59 | Reserved |
| 5a | LSB of Sub-Device ID |
| 5b | MSB of Sub-Device ID |
| 5c | LSB of Sub-Vendor ID |
| 5d | MSB of Sub-Vendor ID |
| 5e-6f | Reserved |
| 70 | Network ID index: to indicates the starting address of Network ID in length of continuous 6 bytes. The content of this field could be in the range of 00-04h, or 10-14h, or 21-24h, or 31-34h |
| 71-76 | Reserved, and should be set to 0 |
| 77 | LED option: The content of this field will be read by driver for LED setting<br>Bit0:CSR9 Bit 28; LED0SEL<br>Bit1:CSR9 Bit 29; LED1SEL |
| 78-79 | Reserved, and should be set to 0 |
| 7a | LSB of Device ID |
| 7b | MSB of Device ID |
| 7c | LSB of Vendor ID |
| 7d | MSB of Vendor ID |
| 7e-7f | Reserved, and should be set to 0 |

## APPENDIX B: MAGNETIC COMPONENTS

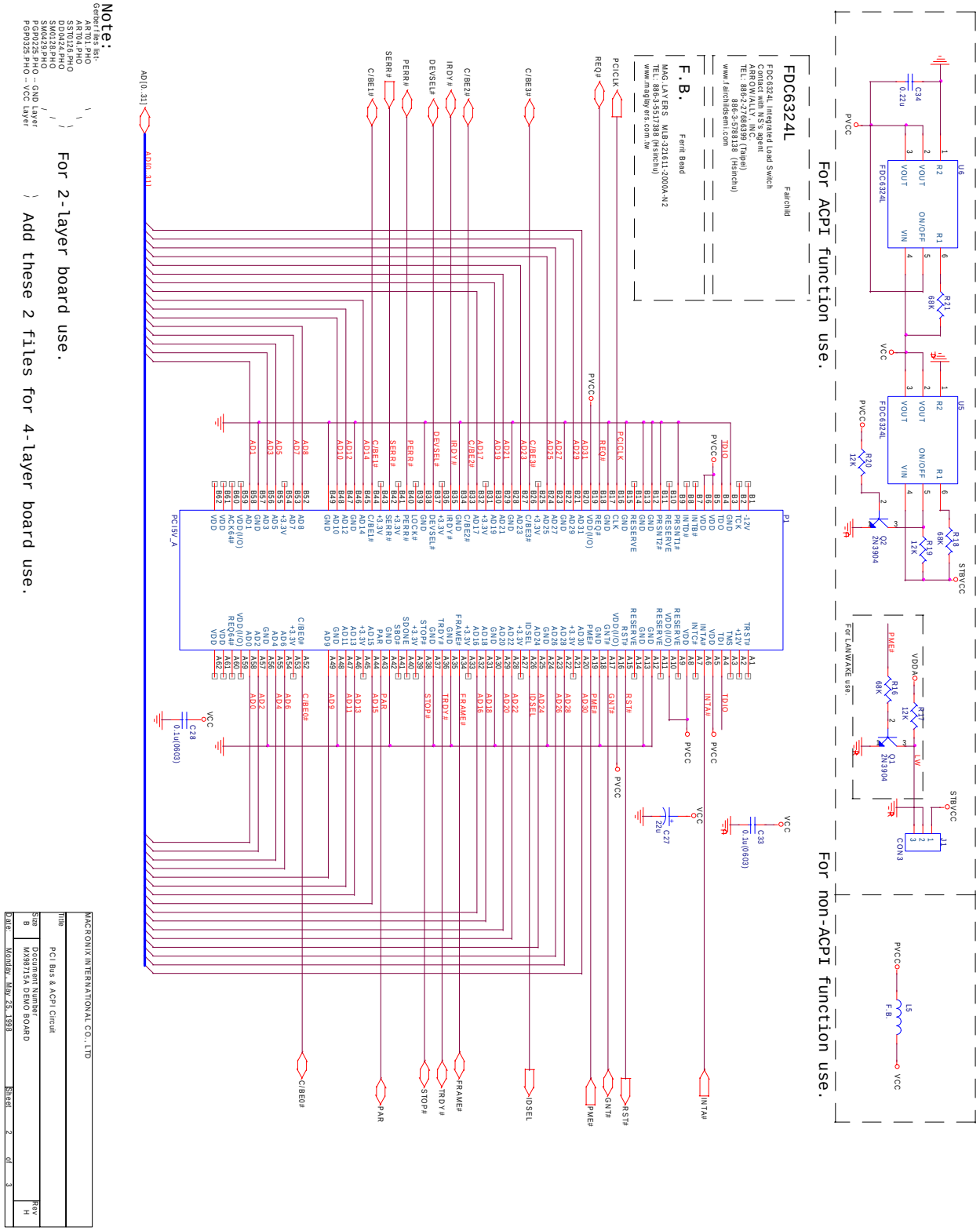
| | | |
|---|---|---|
| 1 BASIC ELECTRICAL SPECIFICATION | | |
| Turn Ratio | Transmit | 1:1 |
| | Receive | 1:1 |
| OCL | 350uH min measured between 0 and 70℃ with a 0.1V rms, 100KHz signal at a DC. bias between 0 and 8mA. | |
| LL | 0.4uH Max at >1MHz | |
| Cww | 18pF Max | |
| DCR | 0.9W Max per winding | |
| Isolation Resistance | not less than 1GW @ 2000V rms | |
| Isolation Voltage | 2000V rms Min @ 60Hz for 1 min | |
| Rise/Fall Time | 3ns Min 4ns Max | |
| Insertion Loss (100 KHz to 100 MHz) | -1.1 dB Max | |
| CMDR & DCMR (100 KHz to 80 MHz) | 38 dB Min | |
| Cross Talk (100KHz to 80 MHz) | -38 dB Max | |
| 2 REFERENCE VENDORS | | |
| Vendor | Part No | |
| Valor | ST6118 (PT4171S) | |
| PE | PE68515 | |
| BelFuse | S558-5999-15 | |
| Delta | LF8200 | |
| Taimic | HSIP-002 | |

## APPENDIX C: THE B.O.M. LIST OF MX98715 DEMO BOARD

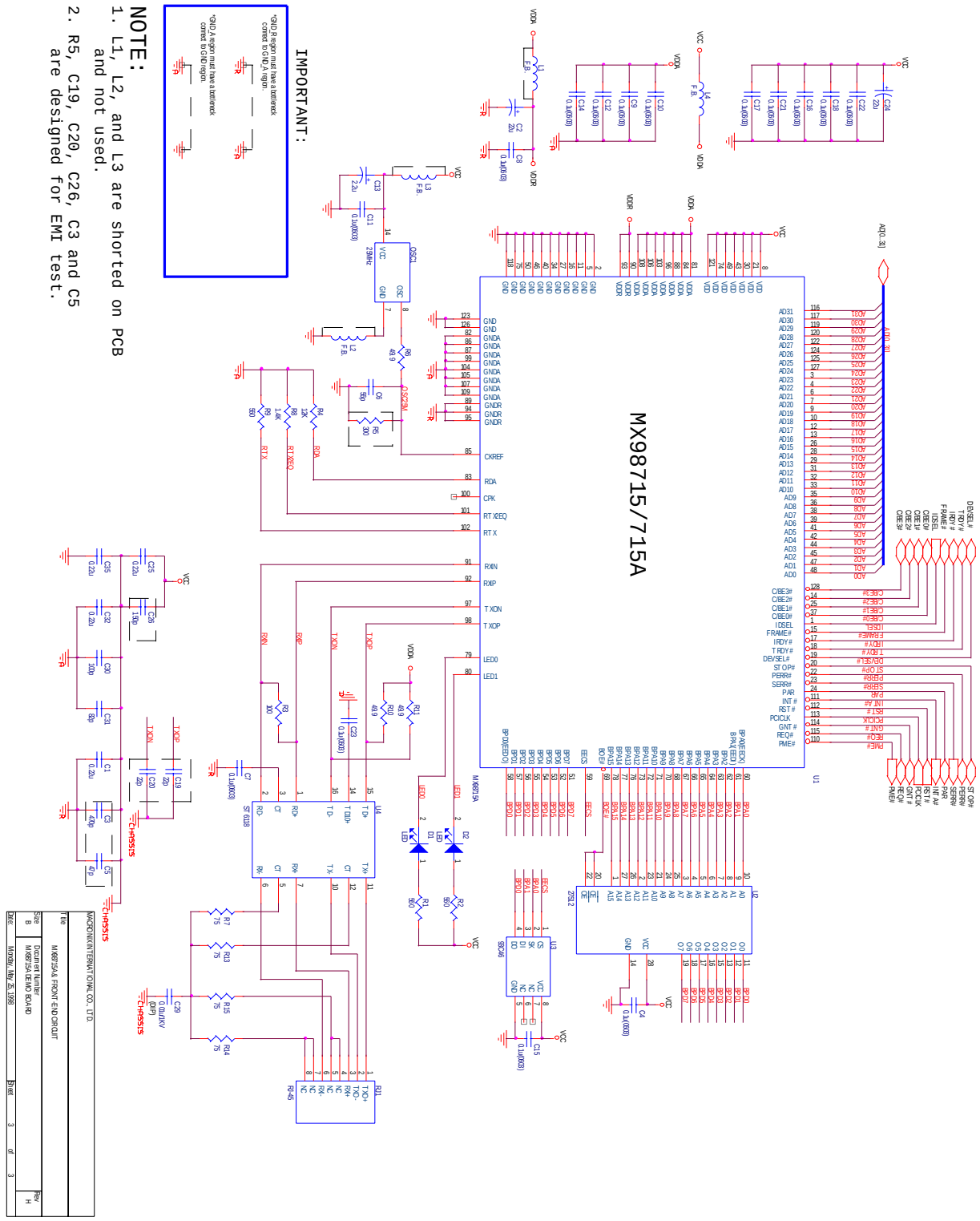
| ITEM | QUANTITY | REFERENCE | PART |
|---|---|---|---|
| 1 | 5 | C1, C25, C32, C34, C35 | 0.22u |
| 2 | 3 | C2, C24, C27 | 22u (DIP) |
| 3 | 17 | C4, C7, C8, C9, C10, C11, C12, C14, C15, C16, C17, C18, C21, C22, C23, C28, C33 | 0.1u (0603) |
| 4 | 1 | C6 | 56p |
| 5 | 1 | C13 | 2.2u (DIP) |
| 6 | 1 | C29 | 0.01u/1KV |
| 7 | 1 | C30 | 100p |
| 8 | 1 | C31 | 82p |
| 9 | 2 | D1, D2 | LED |
| 10 | 2 | L4, L5 | F.B. |
| 11 | 1 | OSC1 | 25MHz (Half size) |
| 12 | 3 | R1, R2, R9 | 560 |
| 13 | 1 | R3 | 100 |
| 14 | 1 | R4 | 12K |
| 15 | 3 | R6, R10, R11 | 49.9 |
| 16 | 4 | R7, R13, R14, R15 | 75 |
| 17 | 1 | R8 | 1.4K |
| 18 | 1 | RJ1 | RJ-45 |
| 19 | 1 | U1 | MX98715 |
| 20 | 1 | U2 | 27512 |
| 21 | 1 | U3 | 93C46 |
| 22 | 1 | U4 | ST6188 |

## APPENDIX D: MX98715 APPLICATION SCHEMATIC CIRCUITS

### D.1: PCI BUS INTERFACE



## D.2: MX98715 AND FRONT-END CIRCUITS



## REVISION HISTORY

| Revision No. | Description | Page | Date |
|---|---|---|---|
| 1.4 | To modify resistor R8 value from 1.5KW to 1.4KW | P.2, P,10, P.12 | JUL/10/1998 |
| 1.5 | Revised Magic Packet format of EEPROM | P9 | OCT/09/1998 |

## MACRONIX INTERNATIONAL CO., LTD.

**HEADQUARTERS:**
**TEL:+886-3-578-8888**
**FAX:+886-3-578-8887**

**EUROPE OFFICE:**
**TEL:+32-2-456-8020**
**FAX:+32-2-456-8021**

**JAPAN OFFICE:**
**TEL:+81-44-246-9100**
**FAX:+81-44-246-9105**

**SINGAPORE OFFICE:**
**TEL:+65-747-2309**
**FAX:+65-748-4090**

**TAIPEI OFFICE:**
**TEL:+886-3-509-3300**
**FAX:+886-3-509-2200**

### MACRONIX AMERICA, INC.
**TEL:+1-408-453-8088**
**FAX:+1-408-453-8488**

**CHICAGO OFFICE:**
**TEL:+1-847-963-1900**
**FAX:+1-847-963-1909**

**http : //www.macronix.com**