DM9000/A/9010 PgSROM for Read/Write EEPROM 93C46

Note 2, 10/16/2006

Action:

1. The DAVICOM testing department tested the DM9000A EEPROM Write timing by checking the access time to delay several milliseconds that is more than 3msec.
2. DM9000 serial NIC was function ok to write the EEPROM 93C/LC46, but slowly.

Result: (10/13'06)

It is necessary to wait 3msec for stable writing words continuously into DM9000s EEPROM 93C/LC46, after detecting the self-clear bit0 cleared. So, it takes more than 0.2 second if re-writing all of 64-word word-by-word totally in EEPROM 93C/LC46. But, it is un-necessary to wait for reading continuously from DM9000s EEPROM 93C/LC46, after detecting the self-clear bit0 (ERRE in EPCR) cleared. The program PgSROM9A.exe was examined all passed, to test the EEPROM 93C/LC46 of Atmel/ Micro-chip/ ST/ ATC/ CSI brands, for clock 0.375MHz.

Sample code:

The EEPROM-Write routine (delay 3msec ok) as follows,

```
int eeprom_write(int offset, unsigned char Value)
{
   unsigned int    tmpv, i, j = 0;
   iow(IOport, 0x0c, offset);      /* issue EEPROM address into EPAR REG. 0c */
   iow(IOport, 0x0E, (Value >> 8) & 0xff);
   iow(IOport, 0x0D, Value & 0xff);
   iow(IOport, 0x0B, 0x0);                  /* clear command first */
   tmpv = outp(IOport + 4, 0x12);           /* issue EEPROM+WRITE command */
   do { /* delay 100* 0.1usec, per cycle, for EECK clock 0.375Mhz */
         for (i = 0; i < 10; i ++) { ;}
         tmpv = inp(IOport + 4);
   }
   while ( (j++ <100) && (tmpv & 0x1) );/* wait EEPROM until bit0 cleared ok */
   delay(10); /* wait > 3msec (~10msec) for EEPROM 93LC46 writing stable */
   tmpv = outp(IOport + 4, 0x0);            /* clear EEPROM command again */
   if (j < 100)            CheckOKFlag = C_TRUE;
   else  {           CheckOKFlag = C_FALSE;   break;      }
   return CheckOKFlag;
```

```
}
```

For example,

```
unsigned int Value[64], i;
CheckOKFlag = C_TRUE;
for (i = 0; i < 64; i ++)        /* EEPROM Write 64-word data totally */
   if ( eeprom_write(i, Value[i]) )
   {      CheckOKFlag = C_FALSE;   print("fail");    return CheckOKFlag; break; }
```

The EEPROM-Read routine as follows,

```
unsigned int eeprom_read(int offset)
{
   unsigned int tmpv, i, j = 0;
   iow(IOport, 0x0c, offset);      /* issue EEPROM address into EPAR REG. 0c */
   iow(IOport, 0x0B, 0x0);                  /* clear command first */
   tmpv = outp(IOport + 4, 0x4);            /* issue EEPROM + READ command */
   do { /* delay 100* 0.1usec, per cycle, for EECK clock 0.375MHz */
         for (i = 0; i < 10; i ++) { ;}
         tmpv = inp(IOport + 4);
   }
   while ( (j++ <100) && (tmpv & 0x1) );/* wait EEPROM until bit0 cleared ok */
   tmpv = outp(IOport + 4, 0x0);            /* clear EEPROM command again */
   return ( (ior(IOport, 0x0E) << 8) | ior(IOport, 0x0D) );
}
```

For example,

```
unsigned char Value[128], i;
for (i = 0; i < 64; i ++)
{  /* EEPROM Read 64-word data totally */
   (unsigned int) tmpv = eeprom_read(i);
   Value[i*2+1] = (unsigned char)(tmpv >> 8);
   Value[i * 2] = (unsigned char)tmpv;
}
```