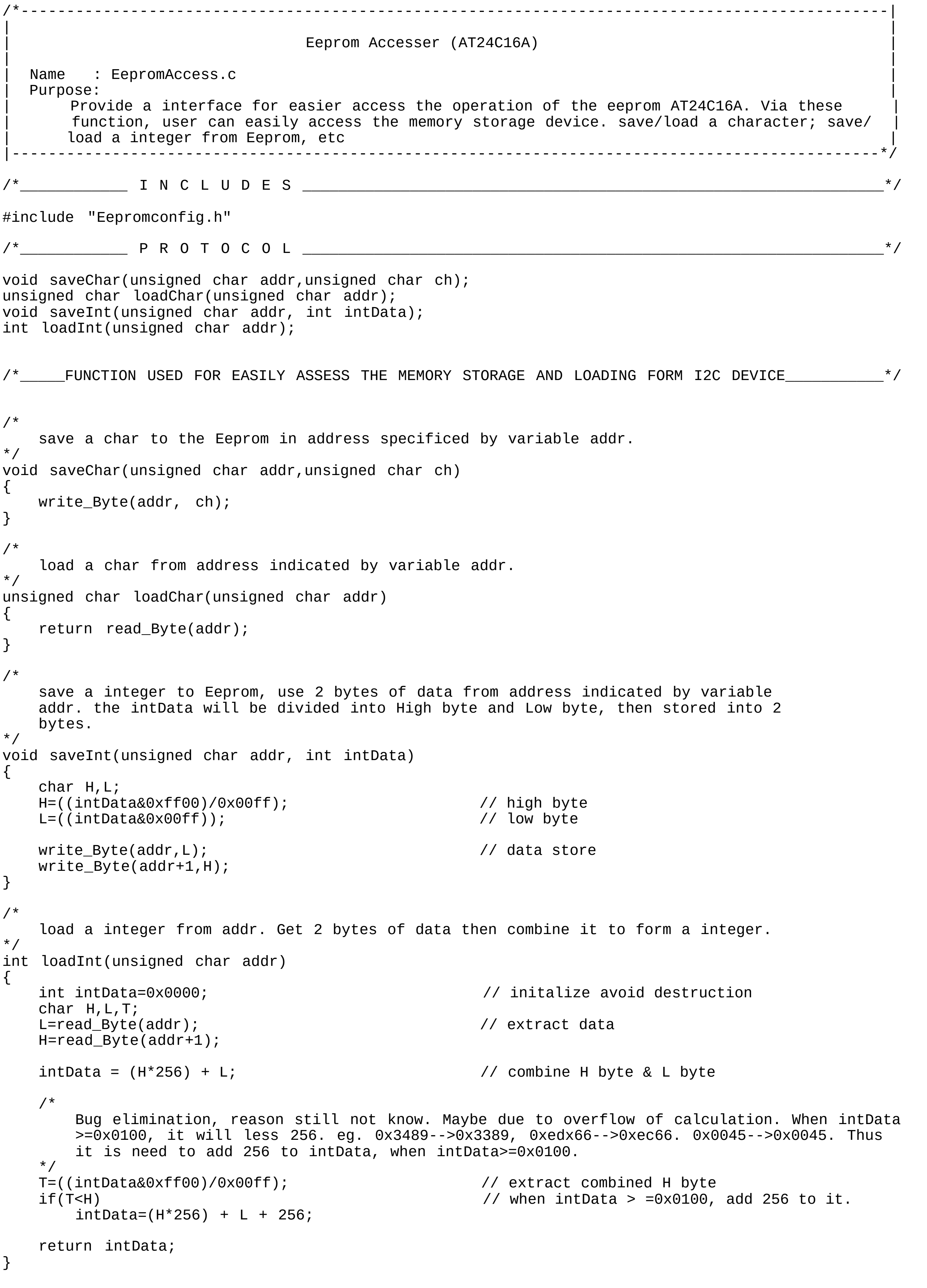
```c
/*-------------------------------------------------------------------------------------------|
|                                                                                            |
|                              8051 Board Demonstration (AT24C16A)                           |
|                                                                                            |
|  Name     : Board.c                                                                        |
|  MCS      : AT89C51                                                                        |
|  Purpose  :                                                                                |
|      A demonstration program to access the memory storage device AT24C16A. The character and   |
|      integer storage is implement, Eeprom Doctor function is included.                     |
|                                                                                            |

|-------------------------------------------------------------------------------------------*/

/*_____________ I N C L U D E S ___________________________________________________________*/

#include  "Eeprom\EepromConfig.h"


/*_____________ C O M  P O R T ____________________________________________________________*/

// com port with 9600 baud with crystal 11.0592MHz.
void  init_uart(void)
{
    SCON  = 0x50;
    TMOD |= 0x20;
    TH1   = 253;
    TR1   = 1;
    TI    = 1;
}

/*_____________ M A I N   F U N C T I O N ______________________________________________________*/

/*
    A demonstration program to access the memory storage device AT24C16A. The character and integer
    storage is implement, Eeprom Doctor function is included.
*/
void  main(void)
{
    char cmd, ch;
    int i;
    init_uart();                                        // 9600 baud @ 11.0592MHz
    printf("Eeprom Access Demo");                       // title
    printf("\n1. Save a Char"                           // menu
           "\n2. Load a Char"
           "\n3. Save a Int"
           "\n4. Load a Int"
           "\nd. Eeprom Doctor"
           );

    while(1){
        printf("\n[Enter Selection :]");
        scanf("%c",&cmd);
        switch(cmd)
        {
            case '1':   printf("\nchar? :");            // save a char
                        scanf("%c",&ch);
                        saveChar(0x01,ch);
                        break;
            case '2':   ch = loadChar(0x01);            // load a char
                        printf("\nchar is : %c",ch);
                        break;
            case '3':   printf("\nInt? : ");            // save a integer
                        scanf("%d",&i);
                        saveInt(0x01,i);
                        break;
            case '4':   i=loadInt(0x01);                // load a integer
                        printf("\nInt is : %d",i);
                        break;
            case 'd':   eepromCheck();                  // call doctor
                        break;
        }
    }
}
```

```
/*-------------------------------------------------------------------------------------|
| Name: config.h                                                                       |
| Purpose:                                                                             |
| config the hardware setting, define the hardware control pin with a notation name    |
| select the require lib for the MPU, and define the protocal for the program access   |
|-------------------------------------------------------------------------------------*/
#include <reg51.h>
#include <stdio.h>

/*____________ H A R D W A R E  _________________________________________________________________*/

sbit    SCL = P0^4;                          // i2c clock pin
sbit    SDA = P0^3;                          // i2c data pin

/*____________ M A C R O S ______________________________________________________________________*/

#define setSCL       SCL=1;while(SCL!=1);
#define clrSCL      SCL=0;
#define setSDA      SDA=1;
#define clrSDA      SDA=0;

#define PAGE_SIZE       7                    // max no. of page size for read/ write
#define SLAVE_ADDRESS   0x50                 // the address of the i2c slave

/*____________ P R O T O C O L _________________________________________________________________*/

// EepromBasic.c
void sendStart(void);
void sendStop(void);
void sendByte(unsigned char);
unsigned char getByte(void);
void sendSlaveAddress(unsigned char b);
void waitAck(void);
void masterAck(void);
void masterNoAck(void);
void sendBitIndicateWrite(void);
void sendBitIndicateRead(void);

// EepromAdvanced.c
void write_Byte(unsigned char eepromAddr, eepromData);
void write_Page(unsigned char eepromAddr, unsigned char wrbuf[PAGE_SIZE]);
unsigned char read_Byte(unsigned char eepromAddr);

// EepromCheck.c
void eepromCheck(void);

// EepromAccess.c
void saveChar(unsigned char addr,unsigned char ch);
unsigned char loadChar(unsigned char addr);
void saveInt(unsigned char addr, int intData);
int loadInt(unsigned char addr);
```

```
/*------------------------------------------------------------------------------------|
| Name: Eeprombasic.c                                                                 |
| Purpose:                                                                            |
| provide the basic level control for the conbined advanced function. Eg, send a start|
| notation, stop notation, send a slave address, byte and ackledgement.               |
|------------------------------------------------------------------------------------*/
#include "Eepromconfig.h"

// a start notation for the i2c slave
void sendStart(void){
    int i;
    setSDA;
    setSCL;
    clrSDA;
    clrSCL;
    for(i=0;i<100;i++);                         // delay for work
}

// a stop notation for the i2c slave
void sendStop(void){
    int i;
    clrSDA;
    setSCL;
    setSDA;
    clrSCL;
    for(i=0;i<100;i++);                         // delay for work
}

// sends one byte of data to a i2c slave
void sendByte(unsigned char b){
    unsigned char mask;
    mask = 0x80;
    do{                                         // send 8 bits
        if ( b & mask ){
          setSDA;                               // bit is high
        }
        else{
          clrSDA;                               // bit is low
        }
        setSCL;                                 // toggle a clock
        clrSCL;
        mask = mask/2;                          // shift mask right
    }while (mask>0);
}

// gets one byte of data from i2c device
unsigned char getByte(void){
    int i;
    unsigned char rcvdata=0x00;
    unsigned char  mask;                         // variable for getting the reading data

    mask = 0x80;
    do{                                         // store 8 bit data
        setSCL;                                 // negative edge clock data out
        for(i=0;i<50;i++);                      // delay for work
        if (SDA==1)
          rcvdata |= mask;
        clrSCL;
        for(i=0;i<50;i++);                      // delay for work
        mask = mask/2;
    }while (mask>0);
    return rcvdata;
}

// sends lower 7 bit of data to a i2c slave
void sendSlaveAddress(unsigned char b){
    unsigned char mask;
    mask = 0x80;
    b*=2;                                       // shift b left
    do{                                         // send 7 bits
        if ( b & mask ){
          setSDA;                               // bit is high
        }
        else{
          clrSDA;                               // bit is low
        }
        setSCL;                                 // toggle a clock
        clrSCL;
        mask = mask/2;                          // shift mask right
    }while (mask>1);
}

// wait until acknoledgment is receviced from slave
void waitAck(void){
    int i;
    for(i=0;i<50;i++);                              // delay for work
    setSDA;                                         // realease SDA for acknowledge
    setSCL;                                         // send clock for acknowledge
    while(SDA);
    clrSCL;
    for(i=0;i<50;i++);                              // delay for work
}

// master acknoledge by sending a clr bit to slave
void masterAck(void){
    clrSDA;
    setSCL;
    clrSCL;
}

// master disacknoledge by sending a set bit to slave
void masterNoAck(void){
    setSDA;
    setSCL;
    clrSCL;
}

// a bit after the slave address, clr indicate i2c write
void sendBitIndicateWrite(void){
    clrSDA;
    setSCL;
    clrSCL;
}

// a bit after the slave address, set indicate i2c read
void sendBitIndicateRead(void){
    setSDA;
    setSCL;
    clrSCL;
}
```

```c
/*---------------------------------------------------------------------------------------|
| Name: advanced.c                                                                       |
| Purpose:                                                                               |
| provide the advanced function of the i2c eeprom. write byte, read byte, write page     |
| and sequential read.(PS: function read_Byte is call by value, read_Page is call by     |
| address)                                                                               |
|---------------------------------------------------------------------------------------*/
#include "Eepromconfig.h"

/*
    write a byte to i2c devices.
*/
void write_Byte(unsigned char eepromAddr, wrData)
{
    sendStart();                             // start
    sendSlaveAddress(SLAVE_ADDRESS);         // 7 bit slave address
    sendBitIndicateWrite();                  // a bit for write mode
    waitAck();
    sendByte(eepromAddr);                    // eeprom address
    waitAck();
    sendByte(wrData);                        // a data
    waitAck();
    sendStop();                              // byte written completed
}

/*
    write a page to i2c devices, eepromAddr indicate the starting address for write
    and eepromAddr + PAGE_SIZE is the ending adddress. the data for write is store in
    array wrbuf[]. the array is pass by address that the caller determined.
*/
void write_Page(unsigned char eepromAddr, unsigned char wrbuf[PAGE_SIZE])
{
    unsigned int i;
    sendStart();                             // start notation
    sendSlaveAddress(SLAVE_ADDRESS);         // 7 bit slave address
    sendBitIndicateWrite();                  // a bit for write mode
    waitAck();
    sendByte(eepromAddr);                    // eeprom address
    waitAck();
    for(i=0;i<PAGE_SIZE;i++)                 // send the data for write
    {
        sendByte(wrbuf[i]);                  // some data
        waitAck();
    }
    sendStop();                              // page written completed
}

/*
    read a byte in a particular address of eeprom memory, and return a readed value
    to the caller.
*/
unsigned char read_Byte(unsigned char eepromAddr)
{
    unsigned char rcvdata=0x00;

    /* set the ptr of eeprom address */
    sendStart();                             // set the eeprom location address
    sendSlaveAddress(SLAVE_ADDRESS);         // 7 bit address for slave location
    sendBitIndicateWrite();                  // bit indicated for write mode to
    waitAck();                               // set the memory address of eeprom
    sendByte(eepromAddr);
    waitAck();

    /* current address read */
    sendStart();
    sendSlaveAddress(SLAVE_ADDRESS);
    sendBitIndicateRead();
    waitAck();
    rcvdata=getByte();
    masterNoAck();
    sendStop();                              // read completed
    return rcvdata;
}
```

```
/*-------------------------------------------------------------------------------------------|
|                                                                                             |
|                              Eeprom Doctor (AT24C16A)                                       |
|                                                                                             |
|  Name   : EepromCheck.c                                                                     |
|  Purpose:                                                                                   |
|      Provide a interface for easier to check the operation of the eeprom AT24C16A. Via this  |
|      function, user can check the pin connection; read byte; write byte and write page      |
|      operation of I2C device AT24C16. Extra functions are included for better character display|
|      and show the result of checking.                                                       |
|--------------------------------------------------------------------------------------------*/


/*____________ I N C L U D E S _________________________________________________________________*/

#include "Eepromconfig.h"

/*____________ P R O T O C O L _________________________________________________________________*/

void eeprom_printHex8(unsigned char hexdata);
unsigned char eeprom_hex2ascii(unsigned char bits_data);
void eepromCheck(void);


/*_____FUNCTION USED TO CHECK EEPROM OPERATION__________________________________________________*/

/*
    provide a interface for easier to check the operation of the eeprom.
*/
void eepromCheck(void)
{
    unsigned char rcvdata;
    unsigned char databuf[PAGE_SIZE]={0x01,0x02,0x03,0x04,0x05,0x06,0x07};
    char cmd;
    int i;

    printf("\nEeprom Doctor");               // title
    printf("\n1. setSDA"                     // menu
           "\n2. clrSDA"
           "\n3. setSCL"
           "\n4. clrSCL"
           "\np. write page from address 0x30-0x36"
           "\nw. write byte from address 0x30-0x36"
           "\nr. read byte from address 0x30-0x36"
           );

    while(1)
    {
        printf("\n[Enter selection : ]");
        scanf("%c",&cmd);

        switch(cmd)
        {
            case '1':   printf("\nsetSDA"); setSDA; break;          // pin check.
            case '2':   printf("\nclrSDA"); clrSDA; break;
            case '3':   printf("\nsetSCL"); setSCL; break;
            case '4':   printf("\nclrSCL"); clrSCL; break;

            case 'p':   printf("\nwrite page");                     // page write
                        write_Page(0x30,databuf);
                        printf("....ok!");
                        break;

            case 'w':   printf("\nwrite Byte");                     // byte write
                        write_Byte(0x30,0x12);
                        write_Byte(0x31,0x34);
                        write_Byte(0x32,0x56);
                        write_Byte(0x33,0x78);
                        write_Byte(0x34,0x90);
                        write_Byte(0x35,0xab);
                        write_Byte(0x36,0xcd);
                        printf("....ok");
                        break;

           case 'r':    printf("\nread Byte\n");        // byte read
                        for(i=0;i<7;i++)
                        {
                            rcvdata=read_Byte(0x30+i);
                            eeprom_printHex8(rcvdata);
                            printf("  ");
                        }
                        break;
        }//endofswitch
    }//endofwhile
}



/*_____EXTRA FUNCTION FOR PRINTING HEX DATA ____________________________________________________*/

/*
    print 8 bit hex data
*/
void eeprom_printHex8(unsigned char hex8)
{
    char hex8H,hex8L;
    hex8L = eeprom_hex2ascii(hex8&0x0f);                 // convert low byte
    hex8H = eeprom_hex2ascii((hex8&0xf0)/0x0f);          // convert high byte
    putchar(hex8H);                                      // print out
    putchar(hex8L);
}

/*
    convert the hex data(4 bits) to ascii code(8 bits)
*/
unsigned char eeprom_hex2ascii(unsigned char bits_data)
{
    if(bits_data >=0x0f)                                 // all invalid data return 'F'
    {
        return 0x46;
    }
    if (bits_data >=0x00 & bits_data <=0x09)             // return '0'-'9'
    {
        bits_data +=0x30;
        return bits_data;
    }
    if (bits_data >=0x0a & bits_data <=0x0f);            // return 'A'-'F'
    {
        bits_data +=0x37;
        return bits_data;
    }
}
```



```
/*---------------------------------------------------------------------------------------|
|                                                                                        |
|                              Eeprom Accesser (AT24C16A)                               |
|                                                                                        |
|  Name   : EepromAccess.c                                                               |
|  Purpose:                                                                              |
|      Provide a interface for easier access the operation of the eeprom AT24C16A. Via these     |
|      function, user can easily access the memory storage device. save/load a character; save/  |
|      load a integer from Eeprom, etc                                                   |
|---------------------------------------------------------------------------------------*/

/*____________ I N C L U D E S _________________________________________________________*/

#include "Eepromconfig.h"

/*____________ P R O T O C O L _________________________________________________________*/

void saveChar(unsigned char addr,unsigned char ch);
unsigned char loadChar(unsigned char addr);
void saveInt(unsigned char addr, int intData);
int loadInt(unsigned char addr);


/*_____FUNCTION USED FOR EASILY ASSESS THE MEMORY STORAGE AND LOADING FORM I2C DEVICE___________*/


/*
    save a char to the Eeprom in address specified by variable addr.
*/
void saveChar(unsigned char addr,unsigned char ch)
{
    write_Byte(addr, ch);
}

/*
    load a char from address indicated by variable addr.
*/
unsigned char loadChar(unsigned char addr)
{
    return read_Byte(addr);
}

/*
    save a integer to Eeprom, use 2 bytes of data from address indicated by variable
    addr. the intData will be divided into High byte and Low byte, then stored into 2
    bytes.
*/
void saveInt(unsigned char addr, int intData)
{
    char H,L;
    H=((intData&0xff00)/0x00ff);                        // high byte
    L=((intData&0x00ff));                               // low byte

    write_Byte(addr,L);                                 // data store
    write_Byte(addr+1,H);
}

/*
    load a integer from addr. Get 2 bytes of data then combine it to form a integer.
*/
int loadInt(unsigned char addr)
{
    int intData=0x0000;                                 // initalize avoid destruction
    char H,L,T;
    L=read_Byte(addr);                                  // extract data
    H=read_Byte(addr+1);

    intData = (H*256) + L;                              // combine H byte & L byte

    /*
        Bug elimination, reason still not know. Maybe due to overflow of calculation. When intData
        >=0x0100, it will less 256. eg. 0x3489-->0x3389, 0xedx66-->0xec66. 0x0045-->0x0045. Thus
        it is need to add 256 to intData, when intData>=0x0100.
    */
    T=((intData&0xff00)/0x00ff);                        // extract combined H byte
    if(T<H)                                             // when intData > =0x0100, add 256 to it.
        intData=(H*256) + L + 256;

    return intData;
}
```