8051 Codes

Example codes (in C) for the classical Intel-8051 (mcu51) microcontroller core. Using my1code51 library. Tested using sdcc open source compiler and stc12 device.

Codes for NMK322

Codes to test hardware modules available for our NMK322 Microcontroller lab.

Code: IR module and Ultrasonic sensor module

Testing IR module and HC-SR04 ultrasonic sensor module.

nmk322_test0.c

```
/*-------------------------------------------------------------------------
-------*/
#include "uart.h"
#include "timer.h"
#define FPSIZE 2
#include "utils_float.h"
#define _LCD_4BIT_INTERFACE_
/** DB4-DB7 => P0 (UPPER NIBBLE) */
/** E => P0.2 : NEEDS -ve EDGE */
/** R/W => P0.1 : 0=WR 1=RD */
/** RS => P0.0 : 0=CMD 1=DAT */
#include "textlcd.h"
/*-------------------------------------------------------------------------
-------*/
/**
 * Testing IR module and HC-SR04 ultrasonic sensor module
 * - display on both uart and lcd
**/
/*-------------------------------------------------------------------------
-------*/
MY1SBIT(IRMOD,PIN10);
MY1SBIT(TRIG,PIN16);
MY1SBIT(ECHO,PIN17);
/*-------------------------------------------------------------------------
-------*/
MY1SBIT(TEST_IR,PIN14);
MY1SBIT(TEST_HC,PIN15);
/*-------------------------------------------------------------------------
-------*/
#define FLAG_IR 0x01
```

```
#define FLAG_HC 0x02
/*-------------------------------------------------------------------------
-------*/
void main(void) {
    unsigned char wait;
    unsigned int tval,loop, flag;
    float fval, dist;
    char buff[16];
    uart_init();
    uart_puts("\r\nTESTING NMK322 STUFFS\r\n");
    lcd_init();
    lcd_goto_line1();
    lcd_puts("TESTKIT 4 NMK322");
    timer_init();
    flag = 0;
    while (1) {
        if (TEST_IR==0) {
            if ((flag&FLAG_IR)==0) {
                lcd_goto_line1();
                lcd_puts("TEST 4 IR MODULE");
                uart_puts("\r\nTESTING IR MODULE\r\n");
                flag = FLAG_IR;
            }
            lcd_goto_line2();
            lcd_puts("                ");
            lcd_goto_line2();
            lcd_puts("Waiting... ");
            uart_puts("\r\nWaiting... ");
            while (IRMOD); // outputs logic low when obstacle detected
            lcd_puts("|*");
            uart_puts("|*");
            while (!IRMOD);
            lcd_puts("|");
            uart_puts("|\r\n");
            for (loop=65000;loop;loop--);
        }
        if (TEST_HC==0) {
            if ((flag&FLAG_HC)==0) {
                lcd_goto_line1();
                lcd_puts("TEST HC-SR04 MOD");
                uart_puts("\r\nTESTING HC-SR04 MODULE\r\n");
                flag = FLAG_HC;
            }
            lcd_goto_line2();
            lcd_puts("                ");
            lcd_goto_line2();
            timer_prep(0);
            TRIG = 1;
            for (wait=10;wait;wait--); // arounf 10us?
            TRIG = 0;
            while (!ECHO); timer_exec();
```

```
            while (ECHO); timer_stop();
            tval = ((unsigned int)TH0<<8)|TL0;
            fval = (float)tval * 1.085; // in us
            dist = fval / 58.0; // in cm
            lcd_puts("Dist:");
            uart_puts("-- Distance=");
            float2str(buff,dist);
            lcd_puts(buff);
            lcd_puts("cm");
            uart_puts(buff);
            uart_puts(" cm\r\n");
            for (loop=65000;loop;loop--);
        }
    }
}
/*---------------------------------------------------------------------------
-------*/
```

Code: BT module and servo

Testing HC06 bluetooth module , TowerPro MG996R servo.

nmk322_test1btservo.c

```
/*---------------------------------------------------------------------------
-------*/
/**
 * Testing HC06 bluetooth module , TowerPro MG996R servo
 * - note: servo pin (1|orange:PWM)(2|red:vcc)(3|brown:ground)
**/
/*---------------------------------------------------------------------------
-------*/
#define APPTITLE "NMK322 BT/SERVO"
#define BTNAME "nmk322bt"
#define BTPASS "0000"
#include "hc06.h"
#include "hc06_conf.h"
#include "uart.h"
#include "uart_hexascii.h"
#include "timer.h"
#include "utils.h"
#include "oled_ssd1306.h"
#include "cstr_hexascii.h"
/*---------------------------------------------------------------------------
-------*/
#define OLED_ROW_DATA 2
#define OLED_ROW_INFO1 4
```

```
#define OLED_ROW_INFO2 6
/*-------------------------------------------------------------------------
-------*/
#define SERVO_PIN P1_0
#include "servo.h"
/*-------------------------------------------------------------------------
-------*/
_sbit(GO000,PIN15);
_sbit(GO1MS,PIN16);
_sbit(GO2MS,PIN17);
/*-------------------------------------------------------------------------
-------*/
#define BUFFSIZE 128
/*-------------------------------------------------------------------------
-------*/
__xdata char buff[BUFFSIZE];
/*-------------------------------------------------------------------------
-------*/
void uart_show_buff(void) {
    char* pchk = buff;
    while (*pchk) {
        if (pchk[0]<0x20||pchk[0]>0x7e) {
            uart_puts("[0x");
            uart_send_hexbyte(pchk[0]);
            uart_send(']');
        }
        else uart_send(pchk[0]);
        pchk++;
    }
    uart_send('\n');
}
/*-------------------------------------------------------------------------
-------*/
void main(void) {
    unsigned char curr, test;
    GO000 = 1; GO1MS = 1; GO2MS = 1;
    timer_init();
    servo_init();
    i2c_init();
    oled1306_init();
    oled1306_puts(APPTITLE);
    uart_init();
    uart_puts("\n---------------\n");
    uart_puts(APPTITLE);
    uart_puts("\n---------------\n\n");
    hc06_init();
    curr = 3;
    do {
        uart_puts("-- Looking for HC06... ");
        oled1306_set_cursor(OLED_ROW_INFO1,0);
        oled1306_puts("Finding HC06... ");
```

```
        hc06_find();
        if (hc06_wait_ok()==HC06_OK) {
            uart_puts("OK.\r\n");
            oled1306_puts("OK");
            oled1306_set_cursor(OLED_ROW_INFO2,0);
            oled1306_puts("Setting up HC06... ");
            uart_puts("\r\nSet default name & pin\r\n");
            uart_puts("-- Set Name (");
            uart_puts(BTNAME);
            uart_puts("): ");
            hc06_setname(buff,BUFFSIZE);
            uart_puts(buff);
            uart_puts("\r\n");
            timer_delay1s(test,1);
            /* set pass */
            uart_puts("-- Set Pin (");
            uart_puts(BTPASS);
            uart_puts("): ");
            hc06_setpin(buff,BUFFSIZE);
            uart_puts(buff);
            uart_puts("\r\n");
            timer_delay1s(test,1);
            oled1306_puts("OK");
            timer_delay1s(test,1);
            oled1306_clear_row(OLED_ROW_INFO1);
            oled1306_clear_row(OLED_ROW_INFO2);
            break;
        }
        uart_puts("timeout.\n");
        timer_delay1s(test,1);
    } while (--curr);
    while (1) {
        if (hc06_peek()) {
            if ((test=hc06_wait(buff,BUFFSIZE))>1) {
                if (buff[0]=='#') {
                    if (test==BUFFSIZE) test--;
                    buff[test] = 0x0;
                    while (buff[test-1]=='\r'||buff[test-1]=='\n') {
                        test--;
                        buff[test] = 0x0;
                    }
                    curr = (unsigned char) str2uint(&buff[1]);
                    if (curr>=5&&curr<=25) {
                        uart_puts("## Turning to (");
                        uart_puts(&buff[1]);
                        uart_puts(")\r\n");
                        oled1306_set_cursor(OLED_ROW_DATA,0);
                        oled1306_puts("ServoTurn: ");
                        oled1306_puts(&buff[1]);
                        servo_turn(curr);
                        timer_delay1s(test,1);
```

```
                        oled1306_clear_row(OLED_ROW_DATA);
                    }
                    else {
                        uart_puts("** Invalid angle (");
                        uart_puts(&buff[1]);
                        uart_puts(")\r\n");
                    }
                }
                else {
                    uart_puts(">> ");
                    uart_puts(buff);
                    uart_puts("\r\n");
                }
            }
        }
        if (GO000==0) {
            oled1306_set_cursor(OLED_ROW_DATA,0);
            oled1306_puts("ServoTurn CENTER");
            servo_turn(15);
            while (!GO000);
            oled1306_clear_row(OLED_ROW_DATA);
        }
        if (GO1MS==0) {
            oled1306_set_cursor(OLED_ROW_DATA,0);
            oled1306_puts("ServoTurn PWM1ms");
            servo_turn(10); // 1ms pwm
            while (!GO1MS);
            oled1306_clear_row(OLED_ROW_DATA);
        }
        if (GO2MS==0) {
            oled1306_set_cursor(OLED_ROW_DATA,0);
            oled1306_puts("ServoTurn PWM2ms");
            servo_turn(20); // 2ms pwm
            while (!GO2MS);
            oled1306_clear_row(OLED_ROW_DATA);
        }
    }
}
/*------------------------------------------------------------------------
-------*/
```

Testing HC06 bluetooth module , TowerPro MG996R servo & Text LCD.

nmk322_test1.c

```
/*------------------------------------------------------------------------
-------*/
/**
```

```
 * Testing HC06 bluetooth module , TowerPro MG996R servo & Text LCD
 * - note: servo pin (1|orange:PWM)(2|red:vcc)(3|brown:ground)
**/
/*-------------------------------------------------------------------------
-------*/
#define BTNAME "nmk322bt"
#define BTPASS "0000"
#include "hc06.h"
#include "uart.h"
#include "timer.h"
#include "utils.h"
/*-------------------------------------------------------------------------
-------*/
/** interface as defined in nmk322 lab module */
#define _LCD_8BIT_INTERFACE_
#define _LCD_PINS_DEFINED_
MY1SFR(LCD_DATA,0xA0);
MY1SBIT(LCD_BUSY,0xA7);
MY1SBIT(LCD_DNC,0x87);
MY1SBIT(LCD_RNW,0x86);
MY1SBIT(LCD_ENB,0x85);
/*-------------------------------------------------------------------------
-------*/
#include "textlcd.h"
/*-------------------------------------------------------------------------
-------*/
#define SERVO_PIN P1_0
#include "servo.h"
/*-------------------------------------------------------------------------
-------*/
MY1SBIT(GO000,PIN15);
MY1SBIT(GO1MS,PIN16);
MY1SBIT(GO2MS,PIN17);
/*-------------------------------------------------------------------------
-------*/
#define BUFFSIZE 128
/*-------------------------------------------------------------------------
-------*/
__xdata char buff[BUFFSIZE];
/*-------------------------------------------------------------------------
-------*/
void main(void) {
    unsigned char curr, test;
    GO000 = 1; GO1MS = 1; GO2MS = 1;
    timer_init();
    servo_init();
    lcd_init();
    lcd_goto_line1();
    lcd_puts("NMK322 SV/LCD/BT");
    uart_init();
    uart_puts("NMK322 SV/LCD/BT\r\n\r\n");
```

```
    hc06_init();
    curr = 3;
    do {
        uart_puts("-- Sending AT... ");
        hc06_find();
        if (hc06_wait_ok()==HC06_OK) {
            uart_puts("OK.\r\n");
            uart_puts("\r\nSet default name & pin\r\n");
            uart_puts("-- Set Name (");
            uart_puts(BTNAME);
            uart_puts("): ");
            hc06_setname(buff,BUFFSIZE);
            uart_puts(buff);
            uart_puts("\r\n");
            timer_delay1s(test,1);
            /* set pass */
            uart_puts("-- Set Pin (");
            uart_puts(BTPASS);
            uart_puts("): ");
            hc06_setpin(buff,BUFFSIZE);
            uart_puts(buff);
            uart_puts("\r\n");
            timer_delay1s(test,1);
            break;
        }
        uart_puts("timeout.\n");
        timer_delay1s(test,1);
    } while (--curr);
    while (1) {
        if (hc06_peek()) {
            if (hc06_wait(buff,BUFFSIZE)>1) {
                if (buff[0]=='#') {
                    curr = (unsigned char) str2uint(&buff[1]);
                    if (curr>=5&&curr<=25) {
                        uart_puts("## Turning to (");
                        uart_puts(&buff[1]);
                        uart_puts(")\r\n");
                        servo_turn(curr);
                    } else {
                        uart_puts("** Invalid angle (");
                        uart_puts(&buff[1]);
                        uart_puts(")\r\n");
                    }
                } else {
                    uart_puts(">> ");
                    uart_puts(buff);
                    uart_puts("\r\n");
                }
            }
        }
        if (GO000==0) {
```

```
            lcd_goto_line2();
            lcd_puts("ServoTurn CENTER");
            servo_turn(15);
            while (!GO000);
            lcd_goto_line2();
            lcd_puts("                ");
        }
        if (GO1MS==0) {
            lcd_goto_line2();
            lcd_puts("ServoTurn PWM1ms");
            servo_turn(10); // 1ms pwm
            while (!GO1MS);
            lcd_goto_line2();
            lcd_puts("                ");
        }
        if (GO2MS==0) {
            lcd_goto_line2();
            lcd_puts("ServoTurn PWM2ms");
            servo_turn(20); // 2ms pwm
            while (!GO2MS);
            lcd_goto_line2();
            lcd_puts("                ");
        }
    }
}
/*-------------------------------------------------------------------------
-------*/
```

Code: RFID module and OLED

Testing RFID (FRC522) and OLED (ssd1306).

nmk322_test2rfid.c

```
/*-------------------------------------------------------------------------
-------*/
#include "uart_hexascii.h"
#include "oled_ssd1306.h"
#include "cstr_hexascii.h"
#include "frc522.h"
#define APPTITLE "NMK322 RFID/OLED"
/*-------------------------------------------------------------------------
-------*/
void main(void) {
    __xdata cstr_t buff; /* default: 64-bytes long */
    unsigned char temp, stat, loop, size;
    unsigned char pdat[FRC522_MAX_RXSIZE], reqa[2];
```

```
    /** initialize */
    cstr_init(&buff);
    uart_init();
    i2c_init();
    oled1306_init();
    spi_init();
    atqa = reqa;
    /* initialize mf contactless card reader */
    /** say something... */
    uart_puts("\n----------------\n");
    uart_puts(APPTITLE);
    uart_puts("\n----------------\n\n");
    oled1306_puts(APPTITLE);
    temp = frc522_init();
    if (!temp||temp==0xff) {
        uart_puts("** Cannot find FRC522 hardware! Aborting!\n");
        oled1306_set_cursor(5,0);
        oled1306_puts("** Hardware Error!");
        hang();
    }
    uart_puts("FRC522 found. Firmware version is 0x");
    uart_send_hexbyte(temp);
    uart_puts(".\n");
    cstr_null(&buff);
    cstr_append(&buff,"FW Vers: 0x");
    cstr_append_hexbyte(&buff,temp);
    oled1306_set_cursor(5,0);
    oled1306_puts(buff.buff);
    /** main loop */
    while (1) {
        stat = frc522_scan(pdat,&size);
        if (stat==FRC522_OK) {
            uart_puts("## TAG(");
            uart_send_hexbyte(stat);
            uart_puts("|");
            uart_send_hexbyte(reqa[0]);
            uart_puts(",");
            uart_send_hexbyte(reqa[1]);
            uart_puts("):");
            cstr_null(&buff);
            for (loop=0;loop<size-1;loop++) { /** UID is size-1 bytes
*/
                uart_send('[');
                uart_send_hexbyte(pdat[loop]);
                cstr_append_hexbyte(&buff,pdat[loop]);
                uart_send(']');
            }
            uart_send('\n');
            oled1306_set_cursor(2,0);
            oled1306_puts("#TAG: ");
            oled1306_puts(buff.buff);
```

```
            loop_delay(3000);
            oled1306_clear_row(2);
        }
        else if (stat!=FRC522_ERROR_NO_TAG&&stat!=FRC522_ERROR_REQ_A) {
            uart_puts("** Scan Failed (0x");
            uart_send_hexbyte(stat);
            uart_puts("):");
            for (loop=0;loop<FRC522_MAX_RXSIZE;loop++) {
                uart_send('[');
                uart_send_hexbyte(pdat[loop]);
                uart_send(']');
            }
            uart_send('\n');
            oled1306_set_cursor(3,0);
            oled1306_puts("*TAG: ERROR!");
            loop_delay(3000);
            oled1306_clear_row(3);
        }
    }
}
/*------------------------------------------------------------------------
-------*/
```

Code: Testing Base Kit

Testing Base Kit (202425s1).

nmk322_basetest.c

```
/*------------------------------------------------------------------------
-------*/
/* override default tick values */
#define TIMER_TICK_LEN TIMER_VAL50MS
#define TIMER_TICK_CNT TIMER_LOOP_1S
/*------------------------------------------------------------------------
-------*/
#include "my1stc51.h"
#include "timer_tick.h"
#include "led7seg.h"
/*------------------------------------------------------------------------
-------*/
void main(void) {
    unsigned char loop, mask, seg7[] = { _7SEGNUM_CC_ };
    P1MODE1(); // better pullup current
    loop = 0; mask = (!P2_7)?0xff:0x00;
    P1 = seg7[loop] ^ mask;
    timer_init();
```

```
    timer_tick_exec();
    while (1) {
        P2_0 = !P3_7;
        P2_1 = !P3_6;
        P2_2 = !P3_5;
        P2_3 = !P3_4;
        if (timer_ticked()) {
            loop++; if (loop==10) loop = 0;
            mask = (!P2_7)?0xff:0x00;
            P1 = seg7[loop] ^ mask;
            timer_tick00();
        }
    }
}
/*-------------------------------------------------------------------------
-------*/
```

Code: gtuc51

Test code for the old GTUC51B001 development board.

gtuc51.c

```
/*-------------------------------------------------------------------------
-------*/
/**
 * Demo program for gtuc51 (with io board)
**/
/*-------------------------------------------------------------------------
-------*/
#include "adc0831.h"
#include "keypad_922.h"
#include "textlcd.h"
#include "utils_float.h"
/*-------------------------------------------------------------------------
-------*/
/**
 * Switch & LED Interface for gtuc51 i/o board
 *   - multiplexed (dual/purpose)
 *   - require jumper link settings!
**/
/** SW0, SW1 => P3.3, P3.2 - LAYOUT ERROR */
__sbit __at (0xB3) SW0;
__sbit __at (0xB2) SW1;
/** LED{0-3} => P3.4-P3.7 */
__sbit __at (0xB4) LED0;
__sbit __at (0xB5) LED1;
```

```
__sbit __at (0xB6) LED2;
__sbit __at (0xB7) LED3;
/** LED{RX,TX} => P3.1, P3.0 - LAYOUT ERROR */
__sbit __at (0xB1) LEDRX;
__sbit __at (0xB0) LEDTX;
/** alias LEDX=-2 and LEDY=-1 (LEFT OF LED0) **/
__sbit __at (0xB1) LEDX;
__sbit __at (0xB0) LEDY;
/*-------------------------------------------------------------------------
-------*/
char display[LCD_MAX_CHAR];
unsigned char lcdi, loop;
float value;
keybyte_t keyin;
adcbyte_t check;
__bit adc, adcgo, left, demo;
/*-------------------------------------------------------------------------
-------*/
#define DEMO_IO 1
#define DEMO_ADC 0
/*-------------------------------------------------------------------------
-------*/
#define LOOP_COUNT 20
/*-------------------------------------------------------------------------
-------*/
/* interrupt service routine for timer0 */
void timer_blink(void) __interrupt TF0_VECTOR {
    TR0 = 0; loop--;
    P1 = ~P1;
    if (loop==0) {
        if (!left) {
            loop = LED0; LED0 = LED1; LED1 = LED2; LED2 = LED3;
            LED3 = LEDX; LEDX = LEDY; LEDY = loop;
        }
        else {
            loop = LED3; LED3 = LED2; LED2 = LED1; LED1 = LED0;
            LED0 = LEDY; LEDY = LEDX; LEDX = loop;
        }
        loop = LOOP_COUNT;
    }
    TH0 = 0x4B; TL0 = 0xFD; TR0 = 1; /** 50ms */
}
/*-------------------------------------------------------------------------
-------*/
/* interrupt service routine for timer1 */
void timer_goadc(void) __interrupt TF1_VECTOR {
    TR1 = 0; TH1 = 0x4B; TL1 = 0xFD;
    if (loop>0) {
        TR1 = 1; /** 50ms */
        loop--;
    }
```

```
    else {
        loop = LOOP_COUNT;
        adcgo = adc;
    }
}
/*-------------------------------------------------------------------------
-------*/
/* interrupt service routine for int1 */
void check_switch0(void) __interrupt IE1_VECTOR {
    left = 0;
    if (lcdi<LCD_MAX_CHAR) {
        lcd_data(0x30);
        lcdi++;
    }
}
/*-------------------------------------------------------------------------
-------*/
/* interrupt service routine for int0 */
void check_switch1(void) __interrupt IE0_VECTOR {
    left = 1;
    if (lcdi<LCD_MAX_CHAR) {
        lcd_data(0x31);
        lcdi++;
    }
}
/*-------------------------------------------------------------------------
-------*/
/* main function */
void main(void) {
    /** initalize stuffs */
    demo = DEMO_IO; /* default... just in case */
    TMOD = 0x11; P1 = 0xFF; P3 = 0xFF;
    lcd_init();
    lcd_goto_line1();
    lcd_puts("8051 Select Demo");
    lcd_goto_line2();
    lcd_puts("[SW0]IO [SW1]ADC");
    /* wait for user key press */
    while (1) {
        if (!SW0) {
            demo = DEMO_IO; /* demo switch, led, keypad and P1 */
            while(!SW0); /** wait until the user press & let go */
            break;
        }
        else if (!SW1) {
            demo = DEMO_ADC; /* demo adc */
            while(!SW1);
            break;
        }
    }
    /* select! */
```

```
    if (demo==DEMO_IO) {
        lcd_goto_line1();
        lcd_puts("MY18051 I/O DEMO");
        lcd_goto_line2();
        lcd_puts("                ");
        lcd_goto_line2();
        /** set timer 0 overflow every 50ms - with interrupt handler */
        loop = LOOP_COUNT; lcdi = 0; left = 0;
        P1 = 0xAA; LED0 = 0; IT0 = 1; IT1 = 1;
        EA = 1; ET0 = 1; EX0 = 1; EX1 = 1;
        TH0 = 0x4B; TL0 = 0xFD; TR0 = 1;
        /** main loop */
        while (1) {
            keyin = key_wait_922();
            if (keyin<10&&lcdi<LCD_MAX_CHAR) { /** numeric key! */
                lcd_data(keyin+0x30);
                lcdi++;
            }
            else if(keyin==0x0F) { /** '#' key! */
                lcd_goto_line2();
                lcd_puts("HASHED!        ");
                lcd_goto_line2();
                lcdi = LCD_MAX_CHAR;
            }
            else if(keyin==0x0E) { /** '*' key! */
                lcd_goto_line2();
                lcd_puts("               ");
                lcd_goto_line2();
                lcdi = 0;
            }
        }
    }
    else {
        lcd_goto_line1();
        lcd_puts("MY18051 ADC DEMO");
        lcd_goto_line2();
        lcd_puts("SW0:ADC, SW1:CLR");
        /* initialize adc */
        adc_init();
        adc = 0; adcgo = 0; EA = 1;
        /** adc status indicator */
        LEDRX = adc; LEDTX = !adc;
        /** main loop */
        while (1) {
            if (!SW0) {
                while (!SW0); /** wait until the user lets go */
                adc = !adc;
                adcgo = adc;
                if (!adcgo) {
                    ET1 = 0; TR1 = 0;
                    lcd_puts("*");
```

```
                }
                LEDRX = adc; LEDTX = !adc;
            }
            else if(!SW1) {
                while (!SW1); /** wait until the user lets go */
                lcd_goto_line2();
                lcd_puts("SW0:ADC, SW1:CLR");
            }
            if (adcgo) {
                adcgo = 0;
                lcd_goto_line2();
                lcd_puts("                ");
                /** read adc */
                check = adc_get_data();
                value = ((float) check / 255.0) * VREF;
                lcd_goto_line2();
                lcd_puts("ADC: ");
                float2str(display,value);
                lcd_puts(display);
                lcd_puts(" (");
                int2str(display,check);
                lcd_puts(display);
                lcd_puts(")");
                /** set timer 1 overflow every 50ms - with interrupt
handler */
                ET1 = 1; TH1 = 0x4B; TL1 = 0xFD; TR1 = 1;
            }
        }
    }
}
/*-------------------------------------------------------------------------
-------*/
```