Coding for 8051

Stuffs related to writing codes for the classical Intel-8051 (mcu51) microcontroller core.

Codes for NMK322

Testing IR module and HC-SR04 ultrasonic sensor module.

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--);
        }
    }
}
/*-------------------------------------------------------------------------
-------*/
```

Testing HC06 bluetooth module and TowerPro MG996R servo.

test1.c

```
/*-------------------------------------------------------------------------
-------*/
#define BTNAME "nmk322bt"
#define BTPASS "0000"
#include "hc06.h"
#include "timer.h"
#include "utils.h"
/*-------------------------------------------------------------------------
-------*/
/**
 * Testing HC06 bluetooth module and TowerPro MG996R servo
 * - note: servo pin (1|orange:PWM)(2|red:vcc)(3|brown:ground)
**/
/*-------------------------------------------------------------------------
-------*/
MY1SBIT(SERVO,PIN10);
MY1SBIT(CHECK,PIN12);
MY1SBIT(GO000,PIN16);
MY1SBIT(GO180,PIN17);
/*-------------------------------------------------------------------------
-------*/
void pwm_cycle(unsigned char wide) {
    unsigned char full;
    full = 200-wide;
    SERVO = 1;
    while (wide) {
        timer_wait(0xffa3); // 0.1ms
        wide--;
    }
    SERVO = 0;
    while (full) {
        timer_wait(0xffa3); // 0.1ms
```

```
        full--;
    }
}
/*-------------------------------------------------------------------------
-------*/
void servo_turn(unsigned char wide) {
    unsigned char tlen;
    tlen = 50; /* times 20ms? */
    while (tlen>0) {
        pwm_cycle(wide);
        tlen--;
    }
}
/*-------------------------------------------------------------------------
-------*/
void main(void) {
    __xdata my1atcmd_t hc06;
    unsigned char curr;
    SERVO = 0; CHECK = 0; GO000 = 1; GO180 = 1;
    uart_init();
    timer_init();
    do {
        hc06_init(&hc06);
    } while (hc06_ok(&hc06)==0);
    hc06_setname(&hc06);
    hc06_setpin(&hc06);
    CHECK = 1;
    while (1) {
        if (uart_peek()) {
            if (hc06_wait(&hc06)==0) {
                if (hc06.buff[0]=='#') {
                    curr = (unsigned char) str2uint(&hc06.buff[1]);
                    if (curr>=5&&curr<=25) {
                        uart_puts("## Turning to (");
                        uart_puts(&hc06.buff[1]);
                        uart_puts(")\r\n");
                        servo_turn(curr);
                    } else {
                        uart_puts("** Invalid angle (");
                        uart_puts(&hc06.buff[1]);
                        uart_puts(")\r\n");
                    }
                } else {
                    uart_puts(">> ");
                    uart_puts(hc06.buff);
                    uart_puts("\r\n");
                }
            }
        }
        if (GO000==0) {
            servo_turn(10); // 1ms pwm
```

```
            while (!GO000);
        }
        if (GO180==0) {
            servo_turn(20); // 2ms pwm
            while (!GO180);
        }
    }
}
/*---------------------------------------------------------------------------
-------*/
```

Code: relay

Testing basic relay.

relay.c

```
/*---------------------------------------------------------------------------
-------*/
#include "timer.h"
#include "uart.h"
/*---------------------------------------------------------------------------
-------*/
MY1SBIT(DRIVE,PIN20);
MY1SBIT(INPUT,PIN10);
/*---------------------------------------------------------------------------
-------*/
void main(void) {
    unsigned char wait;
    timer_init();
    uart_init();
    uart_puts("\nTESTING RELAY\n");
    DRIVE = 1; /* active low */
    INPUT = 1;
    while(1) {
        if (INPUT==0) {
            uart_puts("-- Switch ON... ");
            DRIVE = 0;
            wait = 3*TIMER_LOOP_1S;
            do { timer_wait(TIMER_VAL50MS); } while (--wait);
            uart_puts("OFF.\n");
            DRIVE = 1;
            while (!INPUT);
        }
    }
}
/*---------------------------------------------------------------------------
```

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

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;
            }
        }
    }
}
/*-------------------------------------------------------------------------
-------*/
```