NMK32203 - Microcontroller

Note20241017 *This page is being updated - work in progress...*

This course is *Microcontroller*, offered by the Faculty of Electronics Engineering & Technology.

Announcements

[20241017] Updating this for 202425s1 Academic Session.

Lecture Slides

- Lecture 1 - Overview
- Lecture 2 - Architecture
- Lecture 3 - C for 8051
- Lecture 4 - 8051 I/O

Library Codes

Introduction to using 'library' code. Single header include file(s) meant for single source compilation. These are meant to be compiled using Keil 8051 compiler.

I recommend to use my include file (mcu51.h) rather than the regular (reg51.h) because it would be easier for you to use my example codes and library. I wrote and tested them using my own syntax that can be used on both Keil and SDCC compilers.

Library: Serial (UART)

uart.h

```
#ifndef __MY1UART_H__
#define __MY1UART_H__

/* note: this is NOT compatible with keil's printf */
void uart_init(void) {
    SCON = 0x50;
    TMOD &= 0x0F;
    TMOD |= 0x20;
    TH1 = 253;
    TR1 = 1;
}

void uart_send(unsigned char sdat) {
```

```
    SBUF = sdat;
    while (TI==0);
    TI = 0;
}

unsigned char uart_read(void) {
    unsigned char rdat;
    while (RI==0);
    rdat = SBUF;
    RI = 0;
    return rdat;
}

void uart_puts(char* text) {
    while (*text) {
        uart_send(*text);
        text++;
    }
}

#endif /* __MY1UART_H__ */
```

Library: Text LCD

tlcd.h

```
#ifndef __MY1TLCD_H__
#define __MY1TLCD_H__

#define LCD_DATA P2
sbit LCD_RS = P0^7;
sbit LCD_RW = P0^6;
sbit LCD_EN = P0^5;

void tlcd_delay(unsigned char step) {
    unsigned int loop;
    do {
        loop = 1000; while (--loop);
    } while (--step);
}

void tlcd_write(unsigned char cdat) {
    LCD_RS = 1;
    LCD_RW = 0;
    LCD_DATA = cdat;
    LCD_EN = 1;
    LCD_EN = 0;
    tlcd_delay(1);
```

```
}

void tlcd_cmd(unsigned char ccmd) {
    LCD_RS = 0;
    LCD_RW = 0;
    LCD_DATA = ccmd;
    LCD_EN = 1;
    LCD_EN = 0;
    tlcd_delay(2);
}

void tlcd_init(void) {
    tlcd_cmd(0x38); /* 8 bit mode, 1/16 duty, 5x8 font */
    tlcd_cmd(0x38);
    tlcd_cmd(0x38);
    tlcd_cmd(0x06); /* display off */
    tlcd_cmd(0x0c); /* display on, blink cursor on */
    tlcd_cmd(0x01); /* clear display */
}

void tlcd_puts(char* text) {
    while (*text) {
        tlcd_write(*text);
        text++;
    }
}

#endif /* __MY1TLCD_H__ */
```

Library: Timer

timer.h

```
#ifndef __MY1TIMER_H__
#define __MY1TIMER_H__

// useful macro!
#define timer0_init() { TMOD &= 0xF0; TMOD |= 0x01; }
#define timer0_prep(hi,lo) { TH0 = hi; TL0 = lo; }
#define timer0_stop() { TR0 = 0; TF0 = 0; }
#define timer0_exec() { TR0 = 1; }
#define timer0_flag() { timer0_exec(); while (!TF0); }
#define timer0_wait(hi,lo) { timer0_prep(hi,lo); timer0_flag();
timer0_stop(); }
#define timer0_null() { TH0 = 0; TL0 = 0; }
#define timer0_read() ((unsigned int)TH0<<8)|TL0

void timer0_delayms(unsigned int step) {
```

```
    do {
        timer0_wait(0xfc,0x66); // 1 ms delay
    } while (--step);
}

#endif /* __MY1TIMER_H__ */
```

Library: Utilities (str-int conversions)

utils.h

```
#ifndef __MY1UTILS_H__
#define __MY1UTILS_H__

unsigned char uint8_2cstr(char* cstr, unsigned char that) {
    unsigned char size, temp;
    size = 0;
    temp = that/100;
    if (temp) {
        that %= 100;
        cstr[size++] = temp + 0x30;
    }
    temp = that/10;
    if (temp) {
        that %= 10;
        cstr[size++] = temp + 0x30;
    } else if (size)
        cstr[size++] = 0x30;
    cstr[size++] = that + 0x30;
    cstr[size] = 0x0;
    return size;
}

unsigned char int8_2cstr(char* cstr, signed char that) {
    if (that<0) {
        that = -that;
        *cstr = '-';
        cstr++;
    }
    return uint8_2cstr(cstr,(unsigned char)that);
}

unsigned char uint16_2cstr(char* cstr, unsigned int that) {
    unsigned char size, temp;
    unsigned int idiv;
    size = 0; idiv = 10000;
    while (idiv>=10) {
        temp = (unsigned char)(that/idiv);
```

```
        if (temp) {
            that %= idiv;
            cstr[size++] = temp + 0x30;
        } else if (size)
            cstr[size++] = 0x30;
        idiv /= 10;
    }
    cstr[size++] = that + 0x30;
    cstr[size] = 0x0;
    return size;
}

unsigned char int16_2cstr(char* cstr, int that) {
    if (that<0) {
        that = -that;
        *cstr = '-';
        cstr++;
    }
    return uint16_2cstr(cstr,(unsigned int)that);
}

unsigned char cstr_2uint8(char* pstr, unsigned char* pval) {
    unsigned char loop;
    *pval = 0; loop = 0;
    while (pstr[loop]) {
        if (pstr[loop]<0x30||pstr[loop]>0x39) {
            loop = 0;
            break;
        }
        *pval = (*pval*10) + (pstr[loop]-0x30);
        loop++;
    }
    return loop;
}

unsigned char cstr_2uint16(char* pstr, unsigned int* pval) {
    unsigned char loop;
    *pval = 0; loop = 0;
    while (pstr[loop]) {
        if (pstr[loop]<0x30||pstr[loop]>0x39) {
            loop = 0;
            break;
        }
        *pval = (*pval*10) + (pstr[loop]-0x30);
        loop++;
    }
    return loop;
}

unsigned char cstr_2int16(char* pstr, int* pval) {
    unsigned char loop, skip;
```

```
    skip = (pstr[0]=='-') ? 1 : 0;
    loop = cstr_2uint16(&pstr[skip],(unsigned int*)pval);
    if (skip) *pval = -(*pval);
    return loop;
}

#endif /** __MY1UTILS_H__ */
```

Library: Keypad

kpad.h

```
#ifndef __MY1KPAD_H__
#define __MY1KPAD_H__

/* 4x4 keypad interface {R0,R1,R2,R3,C0,C1,C2,C3} */
/* 4x3 keypad interface {C1,R0,C0,R3,C2,R2,R1} */

/** DefaultPins: C0-C3 => P1.4-P1.7, R0-R3 => P1.0-P1.3 */
#ifndef KEY_DATA
#define KEY_DATA P1
#endif
#ifndef _ROWS_AT_UPPER_
#define ROWS_FLAG 0x01
#define COLS_FLAG 0x10
#else
#define ROWS_FLAG 0x10
#define COLS_FLAG 0x01
#endif

/** no-key indicator for keypad */
#define KEY_NOT_CODE 0x10

unsigned char key_scan(void) {
    /** scan for key press */
    unsigned char irow, icol, mask, test;
    mask = ROWS_FLAG;
    for (irow=0;irow<4;irow++) {
        KEY_DATA = ~mask; test = COLS_FLAG;
        for (icol=0;icol<4;icol++) {
            if ((KEY_DATA&test)==0) {
                while ((KEY_DATA&test)==0);
                if (icol==3) return irow+0x0A;
                if (irow==0) return icol+1;
                if (irow==1) return icol+4;
                if (irow==2) return icol+7;
                if (icol==1) return 0;
                return (icol>>1)+0x0e;
```

```
            }
            test <<= 1;
        }
        mask <<= 1;
    }
    return KEY_NOT_CODE;
}

unsigned char key_wait(void) {
    /** wait for key press */
    unsigned char scan;
    while ((scan=key_scan())==KEY_NOT_CODE);
    return scan;
}
```