EKT322 - Embedded System Design

This course is *Embedded System Design*, offered by the School of Computer and Communication Engineering.

The tools you need for the lab sessions are the development environment (get the packed file that comes with SDCC compiler and ASEM-51 assembler) and flash programmer.

Course Synopsis

Firstly, please read this to get a general idea about embedded systems.

This course introduces the common concepts and requirements of an embedded system. Among the topics discussed are power management, system interfaces, system timing, system interrupts, and application development. Consequently, it also covers the design aspects of such systems based on 8051-based microcontrollers. This includes the 8051 core architecture (and some variants), multitasking, multiprocessor design, and the development of hardware/software solution for given specifications.

Course Outcome

1. Able to analyze the concept and requirements of an embedded system
2. Able to design an embedded system based on given specifications
3. Able to develop embedded system software

Academic Session 2012/2013

This semester (semester 2), I will be handling this course for students in the Computer Engineering program.

Announcement 2012/2013

[**201305031057**] The full source code for the Development Board (+I/O) is now available at BitBucket!

[**201302171004**] Welcome to EKT322! First lecture will be on Mon, Feb 18 (1400-1600).

Assessment 2012/2013

	Examinations	Course Work

	Examinations		Course Work		
Total Contribution	70%		30%		
Assessment	Mid-Term Examinations	Final Examinations	Quizzes	Assignments	Mini Project
Contribution	20%	50%	5%	5%	20%

Syllabus 2012/2013

Week	Lecture	Laboratory
20130211-20130215 Week 01	• Unexpected first week • First lecture falls on a holiday	• No lab sessions this week

20130218-20130222 Week 02	• Introduction to Embedded Systems
	  ◦ concepts and requirements
	  ◦ characteristics of embedded system
	  ◦ small-scale embedded system design
	• Microcontrollers
	  ◦ introduction to 8051 microcontrollers
	  ◦ internal architecture and organization
	  ◦ sub-features: internal timer/counter, interrupts, serial interface
		• No lab sessions this week

20130225-20130301 Week 03	• Application Programming
	  ◦ 8051 instruction set and programming model
	  ◦ I/O Port: structure and dual-functions
	  ◦ Timer & counter feature on 8051
		• Lab Work 0: Tools, Assembly

20130304-20130308 Week 04	• Application Programming
	  ◦ assembly to c
	  ◦ c programming
		• Lab Work 1

20130311-20130315 Week 05	• Exercise on Application Design	• Lab Work 2

Week	Lecture	Laboratory

20130318-20130322 Week 06	• Serial Interface
	  ◦ synchronous/asynchronous interface
	  ◦ protocols: RS232, SPI, I2C, USB
	  ◦ 8051 serial port and mode configuration
		• Lab Work 3

20130325-20130329 Week 07	• Interrupts
	  ◦ event sequence
	  ◦ maskable/non-maskable
	  ◦ priorities
	• Exercise on Application Design
		• Lab Work 4

20130401-20130405 Week 08	• Exercise on Application Design • Evaluation: Theoretical Test 1	• LCD-based Application Development • Notice: Finalizing Titles for Mini-Project
20130408-20130412 Semester Break	Have Fun! While doing some revisions!	

20130415-20130419 Week 09	• Design Specifications
	  ◦ power management: idle/sleep modes
	  ◦ battery powered system for mobile applications
	  ◦ using watchdog timer for error recovery (fault tolerance)
		• Mini-project: Development

20130422-20130426 Week 10	• Design Specifications ◦ *(cont.)* • Exercise on Application Design	• Mini-project: Development

20130429-20130503 Week 11	• Multiprocessor System
	  ◦ multiprocessor configurations: master/slave, distributed systems
	  ◦ multiprocessor interface: RS485 protocol
		• Mini-project: Development

Week	Lecture	Laboratory

20130506-20130510 Week 12	• Real-time and Multitasking
	  ◦ introduction to real-time I/O
	  ◦ basic concept of multitasking
		• Mini-project: Development

20130513-20130517 Week 13	• Hardware Interface & Timing
	  ◦ Memory/Peripheral interfacing
	  ◦ timing specifications
	• Exercise on Application Design
		• Mini-project: Development

20130520-20130524 Week 14	• Open-topic / Revision • Exercise on Application Design	• Mini-project: Demo & Viva Sessions
20130527-20130531 Study Week	Study Smart! Only possible if you have worked hard!	

Timetable 2012/2013

Lecture Hours

Group	Location	Day	Time
(3)	DKR1	MON	1400-1600

Lab Hours

Group	Location	Day	Time
(X)	MKP2	MON	0800-1000
(Y)	MKP2	MON	1000-1200
(X)	MKP2	TUE	1300-1500
(Y)	MKP2	TUE	1500-1700

Consultation Hours

Group	Location	Day	Time
($)	ZZZ	YYY	XXXX-XXXX

Note#1 I no longer provide 'official' consultation hours. Just contact me whenever you need to ask something. My past experience providing consultation hours has led me to believe that students today no longer appreciate this.

Academic Session 2011/2012

This semester (semester 1), I will be handling this course for students in the Computer Engineering program.

Announcement 2011/2012

[**201201041353**] Finals Prep-Session is set on Jan 9, 1100-1300 @DKR1

[**201112191218**] Pre-Finals Test : 2030-2200 Dec 20 @Makmal Lukisan Kejuruteraan (Automart)

[**201110251421**] Mid-Term Test : 2030-2200 Nov 1 @Makmal Lukisan Kejuruteraan (Automart)

[**201109071254**] First lecture (one-time session) will be held on TUESDAY, 13 SEP 2011 10am-12pm @DKR3!

Assessment 2011/2012

	Examinations			**Course Work**		
Total Contribution	70%			30%		
Assessment	Theoretical Test 1	Theoretical Test 2	Final Examinations	Quizzes	Assignments	Mini Project
Contribution	10%	10%	50%	5%	5%	20%

Syllabus 2011/2012

Week	**Lecture**	**Laboratory**
20110912-20110916 Week 01	• First lecture falls on a holiday ◦ Planning an earlier 1-hour session for course briefing ◦ ... and maybe some introduction materials if time permits	• No laboratory sessions for this week
20110919-20110923 Week 02	• Introduction to Embedded Systems ◦ concepts and requirements ◦ characteristics of embedded system ◦ small-scale embedded system design • Microcontrollers ◦ introduction to 8051 microcontrollers ◦ internal architecture and organization ◦ sub-features: internal timer/counter, interrupts, serial interface	• Board/component purchase and assembly (if applicable)

Week	Lecture	Laboratory

20110926-20110930 Week 03	• Application Programming
	  ◦ 8051 instruction set and programming model
	  ◦ assembly and C programming language
		• Lab Work 0: Development Environment, Simple Assembly Code

20111003-20111007 Week 04	• Application Programming ◦ *(cont.)*	• Lab Work 1

20111010-20111014 Week 05	• I/O Port
	  ◦ structure and dual-functions
	  ◦ system architecture - external/internal memory, etc.
		• Lab Work 2

20111017-20111021 Week 06	• Timer & Counter
	  ◦ timer/counter feature on 8051
	  ◦ sample applications
	• Exercise on Application Design
		• Lab Work 3

20111024-20111028 Week 07	• Serial Interface
	  ◦ synchronous/asynchronous interface
	  ◦ protocols: RS232, SPI, I2C, USB
	  ◦ 8051 serial port and mode configuration
		• Lab Work 4

20111031-20111104 Week 08	• Interrupts
	  ◦ event sequence
	  ◦ maskable/non-maskable
	  ◦ priorities
	• Exercise on Application Design
	• Evaluation: Theoretical Test 1
		• LCD-based Application Development
		  ◦ Notice: Titles for Mini-Project finalized

20111107-20111111 Semester Break	Have Fun!	

Week	Lecture	Laboratory

20111114-20111118 Week 09	• Design Specifications
	  ◦ power management: idle/sleep modes
	  ◦ battery powered system for mobile applications
	  ◦ using watchdog timer for error recovery (fault tolerance)
		• Mini-project: Development

20111121-20111125 Week 10	• Design Specifications ◦ *(cont.)* • Exercise on Application Design	• Mini-project: Development

20111128-20111202 Week 11	• Multiprocessor System
	  ◦ multiprocessor configurations: master/slave, distributed systems
	  ◦ multiprocessor interface: RS485 protocol
		• Mini-project: Development

20111205-20111209 Week 12	• Real-time and Multitasking
	  ◦ introduction to real-time I/O
	  ◦ basic concept of multitasking
		• Mini-project: Development

20111212-20111216 Week 13	• Hardware Interface & Timing
	  ◦ Memory/Peripheral interfacing
	  ◦ timing specifications
	• Exercise on Application Design
		• Mini-project: Development

20111219-20111223 Week 14	• Open-topic / Revision • Exercise on Application Design	• Mini-project: Demo & Viva Sessions
20111226-20111230 Study Week	Prepare for Battle! …I mean, Exam!	

Timetable 2011/2012

Lecture Hours

Group	Location	Day	Time
(3)	DKR1	FRI	1000-1200

Lab Hours

Group	Location	Day	Time
(X)	MKR1	MON	0800-0900
(Y)	MKR1	MON	1400-1600
(X)	MKR1	TUE	0900-1100
(Y)	MKR1	TUE	1100-1300

Consultation Hours *will be finalized later...*

Group	Location	Day	Time
($)	SME-B10	MON	1000-1100
($)	SME-B10	TUE	1500-1600
($)	SME-B10	WED	0900-1100
($)	SME-B10	WED	1500-1600

Note#1 I have to make 2-hour slot(s) 'by-appointment-only'. If you have any problems with this please come and see me.

Note#2 The consultation hours are 'official' hours that I have allocated specifically for this course. You can still set an appointment at any time that I am available, however, I can't guarantee I can make it!

8051-based Project Resources

Those related to 8051 (& variants...) microcontroller.

Development Tools

- ASEM-51 - (Freeware) 8051 Macro Assembler
- SDCC - (Open Source) Small Devices C Compiler
- MIDE-51 Studio - IDE using ASEM-51 Assembler and SDCC Compiler
- Flash Magic Tool - For programming flash based microcontrollers from NXP
- Keil Compiler Kit - Commercial but can run as Lite/Evaluation software (e.g. 2kb limit for object code)
- Keil Assembler Kit - Commercial but can run as Lite/Evaluation software (e.g. 2kb limit for object code)
- my1nxpflash - Linux flash tool (console-based) for NXP P89V51 microcontrollers

Note: USB-to-Serial Converter based on pl2303 chipset are usually cheaper, but doesn't usually go above the standard baudrate (38400 max?). Get the ones based on FTDI chipset, which is supposed to support non-standard baudrate (up to 460800?).

Timer Baud Rate Calculation(s)

This section requires the math plugin (based on this).

To get the timer value TVAL for baudrate in 8-bit auto-reload mode (Timer mode 2):

$$COUNT=ROUND\left(\frac{2^{SMOD}}{32}\frac{F_{XTAL}}{12}\frac{1}{BAUD_D}\right)$$

$$TVAL=256-COUNT$$

***Note**: the first term $2^{SMOD}/32$ is for UART control and the second $F_{XTAL}/12$ is the frequency of a machine cycle (timer trigger)*

Checking the achieved baudrate:

$$BAUD_A=\frac{2^{SMOD}}{32}\frac{F_{XTAL}}{12}\frac{1}{COUNT}$$

Calculating error:

$$ERROR=\frac{BAUD_A-BAUD_D}{BAUD_D}\times 100\%$$

Using SDCC

Build/Install SDCC

I use this build script (SlackBuild-style) to compile SDCC from source on Linux. DEST_DIR path must be writable by the user (don't need root for this!). An up-to-date version of this script is always available here.

sdcc.build

```
#!/bin/sh

# sdcc.build
# - written by azman@my1matrix.net
# - build script for sdcc (based on my1 SlackBuild scripts)

# INFO FOR PROGRAM TO BE BUILT
THIS_NAME="$(basename $0)"
THIS_PATH="$(cd $(dirname $0);pwd)"
CURR_PATH="$(pwd)"
PROG_NAME=${PROG_NAME:="$(basename $THIS_NAME .${THIS_NAME##*.})"}
PROG_VERS=${PROG_VERS:="3.2.0"} # CHANGE TO WHAT WE NEED
PROG_FULL="$PROG_NAME-$PROG_VERS" # SHOULD BE TOP FOLDER'S NAME
PROG_BALL="$PROG_NAME-src-$PROG_VERS.tar.bz2" # CAN BE OTHER THAN
$PROG_FULL
# PLACE URL ACCORDINGLY!
PROG_PATH="http://downloads.sourceforge.net/project/$PROG_NAME/$PROG_NA
ME/$PROG_VERS/"
```

```
PROG_LOAD="${PROG_PATH}${PROG_BALL}"
# BUILD-RELATED PATH
TEMP_PATH=${TEMP_PATH:="$CURR_PATH/build-$PROG_NAME"}
DEST_PATH=${DEST_PATH:="/home/share/tool/sdcc"}

function do_download()
{
    local fname="$1"
    local label="$2"
    local cpath="$CURR_PATH"
    [[ "$fname" == "" ]] && exit 1 # shouldn't be here?
    [[ "$label" == "" ]] && label="$fname"
    echo -n "Downloading $label:     "
    wget --progress=dot "$fname" 2>&1 | grep --line-buffered "%" | \
     sed -u -e "s,\.,,g" | awk '{printf("\b\b\b\b%4s", $2)}'
    [[ ! -r "$label" ]] &&
        echo "Cannot download source $fname to $cpath/$label!" && exit
1
    echo -ne "\b\b\b\b DONE!\n"
}

[[ ! -r "$PROG_BALL" ]] && do_download $PROG_LOAD $PROG_BALL

rm -rf $DEST_PATH
mkdir -p $TEMP_PATH $DEST_PATH
cd $TEMP_PATH
rm -rf $PROG_FULL $PROG_NAME $PROG_FOLD
tar xvf $CURR_PATH/$PROG_BALL
if [[ -d $PROG_FULL ]]; then
    cd $PROG_FULL
elif [[ -d $PROG_NAME ]]; then
    cd $PROG_NAME
elif [[ -d $PROG_FOLD ]]; then
    cd $PROG_FOLD
else
    echo "Cannot find path '$PROG_FULL'@'$PROG_NAME'@'$PROG_FOLD'!"
    exit 1
fi

./configure --prefix=$DEST_PATH \
 --disable-z80-port --disable-z180-port \
 --disable-r2k-port --disable-r3ka-port --disable-gbz80-port \
 --disable-pic14-port --disable-pic16-port \
 --disable-hc08-port --disable-s08-port \
 --disable-ucsim --disable-ds390-port --disable-ds400-port

make
make install
```

SDCC Makefile

A makefile I use for 8051 projects using SDCC. If you made changes to DEST_DIR in the build script above, you'll have to change the default TOOL_PATH variable. This makefile is available as part of an open source project at BitBucket.

makefile

```
# makefile for sdcc-based projects by azman@my1matrix.net
# - supports multiple source project
# - supports large memory model
# - for single source file in <src> folder, simply do 'make
<source>.hex'
# - memory settings customized for NXP's P89V51RD2

PROJECT = gtuc51main
OBJECTS = gtuc51adc.rel gtuc51key.rel gtuc51lcd.rel gtuc51lib.rel
gtuc51main.rel
TEMPOBJ = *.asm *.lst *.rel *.sym *.lnk *.map *.mem *.rst *.hex *.lk
*.ihx
HEXFILE = $(PROJECT).hex
OUTFILE = $(PROJECT).ihx

DELETE = rm -rf

TOOL_PATH ?= /home/share/tool/sdcc
CC = $(TOOL_PATH)/bin/sdcc
HEX = $(TOOL_PATH)/bin/packihx

CFLAGS += -DUSE_SDCC -I$(TOOL_PATH)/share/sdcc/include
LFLAGS += --code-loc 0x0000 --iram-size 0x0100
LFLAGS += --xram-loc 0x0000 --xram-size 0x300

MMODEL = -mmcs51 --model-small
MODLIB = -L$(TOOL_PATH)/share/sdcc/lib/small

large: MMODEL = -mmcs51 --model-large
large: MODLIB = -L$(TOOL_PATH)/share/sdcc/lib/large

main: this

all: this

this: $(HEXFILE)

large: this

new: clean all

$(HEXFILE): $(OBJECTS)
```

```
    $(CC) $(MMODEL) $(CFLAGS) -o $(OUTFILE) $(OBJECTS) $(LFLAGS)
$(MODLIB)
    $(HEX) $(OUTFILE) > $(HEXFILE)

%.hex: %.ihx
    $(HEX) $< > $@

%.ihx: %.rel
    $(CC) $(MMODEL) $(CFLAGS) -o $@ $^ $(LFLAGS) $(MODLIB) $(OFLAGS)

%.rel: src/%.c src/%.h
    $(CC) $(MMODEL) $(CFLAGS) -c $<

%.rel: src/%.c
    $(CC) $(MMODEL) $(CFLAGS) -c $<

clean:
    -$(DELETE) $(TEMPOBJ)
```

SDCC/KEIL Code Sharing

Given in the header file below is a macro definition that I use to enable the codes I develop using SDCC to be used by those using Keil UV4 compiler. Instead of including this header file, simply copy the macro definition into the source code should also do the trick. This header file is shown here to show an example on how to use the macro.

Updated20130420 *I've included SFR register/bit declarations converted from SDCC's `8051.h` using sed command shown below.*

my1mcu51.h

```
/*-------------------------------------------------------------------------
-------*/
#ifndef __MY1MCU51_H
#define __MY1MCU51_H
/*-------------------------------------------------------------------------
-------*/
/* Macro Definitions - works on sdcc-3.0.0 & keil uvision4 c51 9.02 */
#ifdef USE_SDCC
#define MY1SFR(NAME,ADDR) __sfr __at ADDR NAME
#define MY1SBIT(NAME,ADDR) __sbit __at ADDR NAME
#define INTERRUPT __interrupt
#define NOP __asm__ ("nop")
#else
#define MY1SFR(NAME,ADDR) sfr NAME = ADDR
#define MY1SBIT(NAME,ADDR) sbit NAME = ADDR
#define INTERRUPT interrupt
#define __bit bit
```

```
#define __xdata xdata
#define __code code
#include <intrins.h>
#define NOP _nop_ ()
#endif
/*-------------------------------------------------------------------------
-------*/
#ifdef USE_SDCC_PRINTF
/* SFR Definitions */
MY1SFR(SBUF,0x99);
/* SFR Bit Definitions */
MY1SBIT(RI,0x98); /** SCON.0 */
MY1SBIT(TI,0x99); /** SCON.1 */
/* SDCC printf needs this to be defined */
void putchar(char c)
{
    SBUF = c;
    while (!TI);
    TI = 0;
}
/* SDCC gets (no scanf) needs this to be defined */
char getchar(void)
{
    char c;
    while(!RI);
    c = SBUF;
    RI = 0;
    return c;
}
#else /** revert printf_fast_f to printf if using keil? */
#define printf_fast_f printf
#endif /* USE_SDCC_PRINTF */
/*-------------------------------------------------------------------------
-------*/
/** SPECIAL FUNCTION REGISTER DECLARATIONS */
MY1SFR(P0  ,0x80);
MY1SFR(SP  ,0x81);
MY1SFR(DPL ,0x82);
MY1SFR(DPH ,0x83);
MY1SFR(PCON,0x87);
MY1SFR(TCON,0x88);
MY1SFR(TMOD,0x89);
MY1SFR(TL0 ,0x8A);
MY1SFR(TL1 ,0x8B);
MY1SFR(TH0 ,0x8C);
MY1SFR(TH1 ,0x8D);
MY1SFR(P1  ,0x90);
MY1SFR(SCON,0x98);
/**MY1SFR(SBUF,0x99);*/
MY1SFR(P2  ,0xA0);
MY1SFR(IE  ,0xA8);
```

```
MY1SFR(P3  ,0xB0);
MY1SFR(IP  ,0xB8);
MY1SFR(PSW ,0xD0);
MY1SFR(ACC ,0xE0);
MY1SFR(B   ,0xF0);
/*-------------------------------------------------------------------------
-------*/
/** SFR BIT DECLARATIONS (BIT-ADDRESSABLE SFR) */
/* P0 */
MY1SBIT(P0_0,0x80);
MY1SBIT(P0_1,0x81);
MY1SBIT(P0_2,0x82);
MY1SBIT(P0_3,0x83);
MY1SBIT(P0_4,0x84);
MY1SBIT(P0_5,0x85);
MY1SBIT(P0_6,0x86);
MY1SBIT(P0_7,0x87);
/* TCON */
MY1SBIT(IT0 ,0x88);
MY1SBIT(IE0 ,0x89);
MY1SBIT(IT1 ,0x8A);
MY1SBIT(IE1 ,0x8B);
MY1SBIT(TR0 ,0x8C);
MY1SBIT(TF0 ,0x8D);
MY1SBIT(TR1 ,0x8E);
MY1SBIT(TF1 ,0x8F);
/* P1 */
MY1SBIT(P1_0,0x90);
MY1SBIT(P1_1,0x91);
MY1SBIT(P1_2,0x92);
MY1SBIT(P1_3,0x93);
MY1SBIT(P1_4,0x94);
MY1SBIT(P1_5,0x95);
MY1SBIT(P1_6,0x96);
MY1SBIT(P1_7,0x97);
/* SCON */
/**MY1SBIT(RI  ,0x98);*/
/**MY1SBIT(TI  ,0x99);*/
MY1SBIT(RB8 ,0x9A);
MY1SBIT(TB8 ,0x9B);
MY1SBIT(REN ,0x9C);
MY1SBIT(SM2 ,0x9D);
MY1SBIT(SM1 ,0x9E);
MY1SBIT(SM0 ,0x9F);
/* P2 */
MY1SBIT(P2_0,0xA0);
MY1SBIT(P2_1,0xA1);
MY1SBIT(P2_2,0xA2);
MY1SBIT(P2_3,0xA3);
MY1SBIT(P2_4,0xA4);
MY1SBIT(P2_5,0xA5);
```

```
MY1SBIT(P2_6,0xA6);
MY1SBIT(P2_7,0xA7);
/* IE */
MY1SBIT(EX0 ,0xA8);
MY1SBIT(ET0 ,0xA9);
MY1SBIT(EX1 ,0xAA);
MY1SBIT(ET1 ,0xAB);
MY1SBIT(ES  ,0xAC);
MY1SBIT(EA  ,0xAF);
/* P3 */
MY1SBIT(P3_0,0xB0);
MY1SBIT(P3_1,0xB1);
MY1SBIT(P3_2,0xB2);
MY1SBIT(P3_3,0xB3);
MY1SBIT(P3_4,0xB4);
MY1SBIT(P3_5,0xB5);
MY1SBIT(P3_6,0xB6);
MY1SBIT(P3_7,0xB7);
/* P3 - DUAL PURPOSE */
MY1SBIT(RXD ,0xB0);
MY1SBIT(TXD ,0xB1);
MY1SBIT(INT0,0xB2);
MY1SBIT(INT1,0xB3);
MY1SBIT(T0  ,0xB4);
MY1SBIT(T1  ,0xB5);
MY1SBIT(WR  ,0xB6);
MY1SBIT(RD  ,0xB7);
/* IP */
MY1SBIT(PX0 ,0xB8);
MY1SBIT(PT0 ,0xB9);
MY1SBIT(PX1 ,0xBA);
MY1SBIT(PT1 ,0xBB);
MY1SBIT(PS  ,0xBC);
/* PSW */
MY1SBIT(P   ,0xD0);
MY1SBIT(F1  ,0xD1);
MY1SBIT(OV  ,0xD2);
MY1SBIT(RS0 ,0xD3);
MY1SBIT(RS1 ,0xD4);
MY1SBIT(F0  ,0xD5);
MY1SBIT(AC  ,0xD6);
MY1SBIT(CY  ,0xD7);
/*-------------------------------------------------------------------------
-------*/
/** INTERRUPT VECTOR */
#define IE0_VECTOR 0
#define TF0_VECTOR 1
#define IE1_VECTOR 2
#define TF1_VECTOR 3
#define SI0_VECTOR 4
/*-------------------------------------------------------------------------
```

```
-------*/
#endif /* __MY1MCU51_H */
/*-------------------------------------------------------------------------
-------*/
```

Sed command to change 8051.h by SDCC to the above code,

```
sed -e 's/^__sfr __at (\(.*\)) \([^ ;]*\).*$/MY1SFR(\2,\1);/g' -e 's/^__sbit
__at (\(.*\)) \([^ ;]*\).*;$/MY1SBIT(\2,\1);/g'
```

Updated20130502 *I'm planning to use SDCC specific coding from now on - I'm using sed command shown below to convert the above format back to SDCC.*

```
sed -e 's/^MY1SFR(\(.*\),\(.*\));.*$/__sfr __at (\2) \1;/g' -e
's/^MY1SBIT(\(.*\),\(.*\));.*$/__sbit __at (\2) \1;/g'
```

To change codes (SFR/SBIT declarations only) meant for SDCC to Keil (NOT TESTED!),

```
sed -e 's/^__sfr __at (\(.*\)) \([^ ;]*\).*$/sfr \1=\2;/g' -e 's/^__sbit
__at (\(.*\)) \([^ ;]*\).*;$/sbit \1=\2;/g'
```

SDCC 'Limitation'?

Utilize DJNZ in loops

Instead of doing it the *natural* way,

```
for(loop=10;loop>0;loop--) { ... }
```

... you'd have to go for the *1-on-1 substitution* as such:

```
loop=10;
do {
    // ...
    loop--;
} while(loop>0);
```

I do not know if this issue can be seen when using Keil compiler - I don't use it much. Try it if you're curious!

Update20130522 I have actually seen this same issue when using Keil compiler but using the above recommended code does not solve the issue.

Print floating-point value

To use *%f* in *printf*, you need to re-compile *printf_large.c* with

```
-DUSE_FLOATS=1
```

An easier approach is to use *printf_fast_f*.

'char' Array vs String

I was trying to declare a character array (which what string is, but I really want it as just an array of characters that I want to access individually). So, I declared

```
char hexval[] = { "0123456789ABCDEF" };
```

I knew it'll add a NULL char at the end and effectively having a 17-character array.

However, when I tried to compile the whole program, I get

```
error 47: indirections to different types assignment
```

that did not point to any particular line. After a series of debugging work, I finally realized that the declaration line was the problem and I had to re-declare like this

```
char hexval[] = {
'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F' };
```

That solves the compiler issue.

Known Issue

Note: We have issues with many USB2Serial Cables... (**UPDATE20130316** No longer an issue since we're using FTDI chip)

- autobaud doesn't work? autobaud char is 'U' for 51-based ('?' for ARM-based)
- check common GND between max232 and PC?
- issue: problem with 'cheap' prolific chipset implementations
- solution: lower baudrate (9600 is commonly used)
- ftdi-based cables looks more stable? tested on my Linux machine!
- **UPDATE201203051201**: I bought a 'cheap' cable online - works okay @9600kbps on my Linux machine using self-written flash software!

Sample Codes

These *sample* codes are for the 8051-based system.

Real-Time Clock Project

8051_rtc.asm

```
;*************************************************
; clock app using rtc 8583
;*************************************************

    cpu "8051.tbl"

    ; internal ram bit-addr locations
erb:   equ 070h ; ack error flag
srf:   equ 071h ; stop read flag
sda:   equ 0b0h ; i2c data line (p3.0)
scl:   equ 0b1h ; i2c clock line (p3.1)
ad0:   equ 0b2h ; latch address line 0
ad1:   equ 0b3h ; latch address line 1
sw0:   equ 0b4h ; input switch 0
sw1:   equ 0b5h ; input switch 1
wrs:   equ 0b7h ; write segment latch
    ; internal ram byte locations
tmp:   equ 060h ; temporary storage
tgf:   equ 061h ; sec toggle byte (subb!)
msk:   equ 062h ; port mask for colon blink
tm0:   equ 070h ; control byte
tm1:   equ 071h ; 1/100 byte
tm2:   equ 072h ; second byte
tm3:   equ 073h ; minute byte
tm4:   equ 074h ; hour byte
    ; alias for system reg/port
pt1:   equ 090h ; port 1 - segment data
rgb:   equ 0f0h ; register b

    org 00h
    ljmp start ; skip interrupt vectors area

    ; org 03h ; override with segment table + nulls!!
table: dfb 08h,5bh,22h,12h,51h,14h,04h,5ah    ; org 0bh
    dfb 00h,10h,00h,00h,00h,00h,00h,00h    ; org 13h
    dfb 00h,00h,00h,00h,00h,00h,00h,00h    ; org 1bh
    dfb 00h,00h,00h,00h,00h,00h,00h,00h    ; org 23h
    dfb 00h,00h,00h,00h,00h,00h,00h,00h    ; org 2bh
    dfb 00h,00h,00h,00h,00h,00h,00h,00h    ; org 33h
    dfb 00h,00h,00h,00h,00h,00h,00h,00h

;*************************************************
; main code
;*************************************************

    ; org 3bh
```

```
start:    setb ad0 ; set default latch address 11
    clr wrs ; clear write signal
    clr srf ; clear stop read flag
    mov tgf,#00h ; toggle byte is zero (init)
    mov msk,#00h ; switch on by default - orl
    mov dptr,#table ; init segment display table

    ; local default setting
    mov tm0,#00h ; mode ext osc,nomask,noalm
    mov tm1,#00h ; 1/100 seconds bcd
    mov tm2,#50h ; seconds bcd
    mov tm3,#58h ; minutes bcd
    mov tm4,#92h ; 12h,am,12 hour bcd

    ; init rtc 8583 setting & value
irtc:  call i2cst
    mov a,#0a0h ; slave address (a0=0) write
    call i2cout
    jb erb,irtc ; check acknowledge
    mov a,#00h ; word address (control reg)
    call i2cout
    mov a,tm0 ; mode ext osc,nomask,noalm
    call i2cout
    mov a,tm1 ; 1/100 seconds bcd
    call i2cout
    mov a,tm2 ; seconds bcd
    call i2cout
    mov a,tm3 ; minutes bcd
    call i2cout
    mov a,tm4 ; 12h,am,12 hour bcd
    call i2cout
    call i2csp

    ; main routine
main:  call i2cst
    mov a,#0a0h ; slave address (a0=0) write
    call i2cout
    mov a,#03h ; word address (03-minute)
    call i2cout
    call i2cst ; resend for read
    mov a,#0a1h ;slave address (a0=0) read
    call i2cout
    ; start read
    call i2cin
    mov tm3,a ; minute byte
    setb srf ; last byte
    call i2cin
    mov tm4,a ; hour byte
    call i2csp
    ; now we update!
disp:  setb ad1 ; set latch address 11
```

```
    mov a,tm3
    anl a,#0fh
    movc a,@a+dptr
    orl a,msk
    mov pt1,a
    setb wrs
    clr wrs
    clr ad0 ; set latch address 10
    mov a,tm3
    swap a
    anl a,#0fh
    movc a,@a+dptr
    orl a,msk
    mov pt1,a
    setb wrs
    clr wrs
    clr ad1 ; set latch address 00
    mov a,tm4
    swap a
    anl a,#03h ; for hour take 2-bits only
    movc a,@a+dptr
    orl a,msk
    mov pt1,a
    setb wrs
    clr wrs
    setb ad0 ; set latch address 01
    mov a,tm4
    anl a,#0fh
    movc a,@a+dptr
    orl a,msk
    mov pt1,a
    setb wrs
    clr wrs

    ; check switches
chkb0: jb sw0,chkb1
    jnb sw0,$
    mov a,tm3
    add a,#01h
    da a
    cjne a,#60h,savem
    mov a,#00h
savem: mov tm3,a
    jmp next1
chkb1: jb sw1,chktm
    jnb sw1,$
inch:  mov a,tm4
    anl a,#3fh
    add a,#01h
    da a
    cjne a,#13h,saveh ; only for 12h format!
```

```
    mov a,#01h
saveh: mov tmp,a
    mov a,tm4
    anl a,#0c0h
    orl a,tmp
    mov tm4,a
    ; write to rtc if needed
next1: call i2cst
    mov a,#0a0h ; slave address (a0=0) write
    call i2cout
    mov a,#03h ; word address (03-minute)
    call i2cout
    mov a,tm3 ; minutes bcd
    call i2cout
    mov a,tm4 ; hour bcd
    call i2cout
    call i2csp
    jmp main

    ; monitor tf flag
chktm: call i2cst
    mov a,#0a0h ; slave address (a0=0) write
    call i2cout
    mov a,#00h ; word address (00)
    call i2cout
    call i2cst ; resend for read
    mov a,#0a1h ;slave address (a0=0) read
    call i2cout
    ; start read
    setb srf ; last byte
    call i2cin
    anl a,#01h
    mov tm0,a ; control byte - get tf (lsb) only!
    call i2csp
    ; now we check
    clr c
    subb a,tgf ; look for transition!
    jz chkb0
    mov tgf,tm0
    jnc col0
    mov msk,#00h ; 1->0 transition!
    jmp main
col0:  mov msk,#80h ; 0->1 transition!
    jmp disp

;*************************************************
; subroutines
;*************************************************

; routine i2c to write out start marker
```

```
i2cst: setb scl
    setb sda
    nop ; start condition setup time > 4.7 us
    nop ; for 11.0592MHz xtal, 1 mc = 1.085us
    nop ; 1 nop = 1 mach cycle
    nop
    clr sda
    nop ; start condition hold time > 4.0 us
    nop
    nop
    clr scl
    ret

; routine i2c to write 1 byte out - scl already low

i2cout:    mov rgb,#08h
i2cl1: rlc a
    jnc i2co0
    setb sda
    sjmp i2co1
i2co0: clr sda
i2co1: setb scl
    nop ; scl high time > 4.0 us (low time > 4.7 us)
    nop
    nop
    clr scl
    djnz rgb,i2cl1
    setb sda
    nop
    setb scl
    clr erb
    jnb sda,i2cef ; check acknowledge bit
    setb erb
i2cef: clr scl
    ret

; routine i2c to write out stop marker

i2csp: clr sda
    nop
    setb scl
    nop ; stop condition setup time > 4.0 us
    nop
    nop
    setb sda
    ret

; routine i2c to read 1 byte in

i2cin: mov rgb,#08h
    setb sda ; need this to disable int pulldwn source?
```

```
i2cl2: clr scl
    nop ; scl low time > 4.7 us (high time > 4.0 us)
    nop
    nop
    nop
    setb scl
    clr c
    jnb sda,i2ci0
    cpl c
i2ci0: rlc a
    djnz rgb,i2cl2
    clr scl
    jb srf,i2ci1 ; scl low to vata valid time < 3.4us??
    clr sda ; send acknowledge bit if not last!
i2ci1: clr srf ; reset stop read flag
    nop
    setb scl
    nop
    nop
    nop
    clr scl
    ret

;*****************************************************
    end
;*****************************************************
```

Terminal User Interface

labX.c

```
#include <stdio.h>
/* for setting serial baudrate */
__sfr __at 0X89 TMOD;
__sfr __at 0x98 SCON;
__sfr __at 0x8D TH1;
__sbit __at 0x8E TR1; /** TCON.6 */

/* for serial transmit */
__sfr __at 0x99 SBUF;
__sbit __at 0x98 RI; /** SCON.0 */
__sbit __at 0x99 TI; /** SCON.1 */

/* SDCC printf needs this to be defined */
void putchar(char c)
{
    SBUF = c;
    while (!TI);
```

```
    TI = 0;
}

/* SDCC gets (no scanf) needs this to be defined */
char getchar(void)
{
    char c;
    while(!RI);
    c = SBUF;
    RI = 0;
    return c;
}

#define NAME_SIZE_MAX 16
#define BUFF_SIZE_MAX 4
#define THIS_YEAR 2013

char name[NAME_SIZE_MAX], buff[BUFF_SIZE_MAX];
int year, age;

int convert_int(char* buff)
{
    int calc = 0, length = 0, mult = 1;
    /** get string length */
    while(buff[length]) length++;
    /** browse through the array - from right! */
    while(length>0)
    {
        length--;
        /** check if really numeric */
        if(buff[length]<0x30||buff[length]>0x39)
            return 0;
        /** get digit actual value */
        calc = calc + (buff[length]-0x30)*mult;
        /** prepare next iteration */
        mult = mult * 10;
    }
    return calc;
}

/* main function */
void main()
{
    TMOD = 0x21; SCON = 0x50; TH1 = 253; TR1 = 1;
    while(1)
    {
        /** get user input */
        printf("Enter name [MAX=%d]: ",NAME_SIZE_MAX);
        gets(name);
        printf("Enter your age (Max=99): ");
        gets(buff);
```

```
        /** process year of birth */
        age = convert_int(buff);
        year = THIS_YEAR - age;
        /** display */
        printf("Hi, %s! You were born in %d\n",name,year);
    }
}
```

Useful Support Codes

These codes are meant to be built and executed on desktop machine, but to support embedded development.

Generate sine data for look-up table (LUT)

Creating sine data in a look-up table (LUT) for an embedded system project...

sinedata.c

```
#include <stdio.h>
#include <string.h>
#include <math.h>

#define VALUE_PI 3.14592
#define CHAR_PER_LINE 80

int main(int argc, char* argv[])
{
    int sample_count = 360, sample_bits = 8;
    int sample_peak, loop, test, cols;
    char *pcheck, bufftext[64];
    double temp;

    if(argc>1)
    {
        for(loop=1;loop<argc;loop++)
        {
            /* check for command line options */
            if(argv[loop][0]=='-'&&argv[loop][1]=='-')
            {
                pcheck = &argv[loop][2];
                if(strcmp(pcheck,"sample")==0)
                {
                    loop++;
                    if(loop==argc)
```

```
                    {
                        printf("Using %d as sample
count.\n",sample_count);
                    }
                    else
                    {
                        sample_count = atoi(argv[loop]);
                    }
                }
                else if(strcmp(pcheck,"bits")==0)
                {
                    loop++;
                    if(loop==argc)
                    {
                        printf("Using %d as sample
bitsize.\n",sample_bits);
                    }
                    else
                    {
                        sample_bits = atoi(argv[loop]);
                    }
                }
            }
            else
            {
                printf("Unknown command '%s'!\n",argv[loop]);
            }
        }
    }

    sample_peak = (int)pow(2,sample_bits)-1;
    printf("/* Sample range is between 0 and %d inclusive
*/\n",sample_peak);
    printf("/* Time range is in [0,2PI) */\n");
    printf("int sinedata[%d] = {\n",sample_count);
    for(loop=0,cols=0;loop<sample_count;loop++)
    {
        temp =
sample_peak*(sin((double)loop*2.0*VALUE_PI/sample_count)+1)/2.0;
        test = (int)temp;
        if(temp-(double)test>0.5) test++;
        sprintf(bufftext,"%d",test);
        if(loop<sample_count-1) sprintf(bufftext,"%s,",bufftext);
        cols += strlen(bufftext);
        if(cols>CHAR_PER_LINE) { printf("\n"); cols = strlen(bufftext);
}
        printf("%s",bufftext);
    }
    printf("\n};\n");

    return 0;
```

```
}
```