C Programming on the 8051/8052

```
; Use Timer0 to create a square wave on P1.0
Start:
            MOV   TMOD, #01h          ; 16-bit mode on timer0
            MOV   TH0, #0FEh          ; We start off 2^16 - 500
            MOV   TL0, #0CH
            SETB  TR0                 ; Start timer0
Wait:
            JNB   TF0, Wait           ; Wait for overflow
            CLR   TR0                 ;
            CLR   TF0                 ;
            CPL   P1.0                ;
            JMP   Start               ;
```

```
// Use Timer0 to create a square wave on P1.0
#include <REG52.H>

sbit portbit = P1^0;

void main()
{
   TMOD = 1;                          // 16-bit mode on timer0
   while (1)                          // Loop forever
   {
      TH0 = 0xFE;                     // Start timer at 2^16-500
      TL0 = 0x0C;
      TR0 = 1;                        // Start timer0
      while (TF0 != 1)                // Wait for overflow
         ;
      TR0 = 0;                        // Stop timer
      TF0 = 0;                        // Clear timer overflow flag
      portbit = !portbit;             // Toggle port bit
   }
}
```

Overview

- Assembly versus C
- C data types
- Using C in Keil
- 8051 specific extensions
  - SFRs, bit variables
  - Creating ISRs

C versus Assembly

- Advantages
  - High-level, structured programming language
  - Compiler relieves programmer from some of the hardware details
  - Easier to write large, complex software
  - Programs more readable
- Disadvantages
  - Generally larger machine code
  - Less control and ability to interact with hardware
  - Unclear number of cycles to do something

Square wave, assembly

- Goal: square wave on P1.0
  - High level for 500 ticks, low level for 500 ticks

```
; Use Timer0 to create a square wave on P1.0
Start:
        MOV   TMOD, #01h          ; 16-bit mode on timer0
        MOV   TH0,  #0FEh         ; We start off 2^16 - 500
        MOV   TL0,  #0CH
        SETB  TR0                 ; Start timer0
Wait:
        JNB   TF0, Wait           ; Wait for overflow
        CLR   TR0                 ; Stop timer
        CLR   TF0                 ; Clear timer overflow flag
        CPL   P1.0                ; Toggle port bit
        JMP   Start               ; Repeat forever
```

Square wave, C

```
// Use Timer0 to create a 1kHz square
#include <REG52.H>

sbit portbit = P1^0;

void main()
{
    TMOD = 1;                   // 16-bit mode on timer0
    while (1)                   // Loop forever
    {
        TH0 = 0xFE;             // Start timer at 2^16-500
        TL0 = 0x0C;
        TR0 = 1;                // Start timer0
        while (TF0 != 1)        // Wait for overflow
            ;
        TR0 = 0;                // Stop timer
        TF0 = 0;                // Clear timer overflow flag
        portbit = !portbit;     // Toggle port bit
    }
}
```

Gets a bunch of names for SFRs, etc.

Special 8051 SFR bit data type, ^ specifies which bit

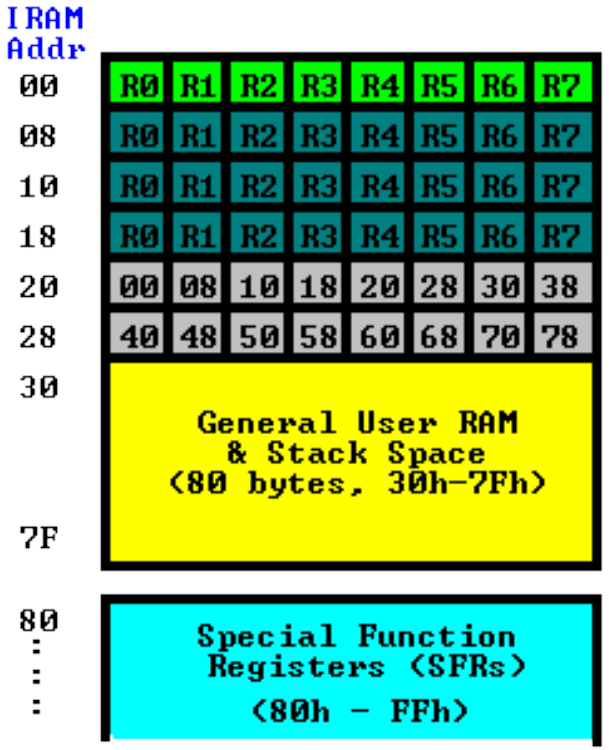
no boolean data type!

Standard data types in 8051 C

Data type	Bytes	Min value	Max value
signed char	1	-128	+127
unsigned char	1	0	255
signed short	2	-32768	+32767
unsigned short	2	0	65535
signed int	2	-32768	+32767
unsigned int	2	0	65535
signed long	4	-2,147,483,648	+2,147,483,647
unsigned long	4	0	4,294,967,295
float	4	±1.175494e-38	±3.402823e+38
double	4	±1.175494e-38	±3.402823e+38

8051 extension types

- 8051 extension types
  - bit
    - 8051 bit addressable memory
    - 20h to 2Fh
  - sbit
    - A bit inside an SFR (e.g. P1.0)
  - sfr
    - Entire byte of an SFR
  - sfr16
    - 16-bit SFRs (e.g. DPTR)



8051 extension types

- 8051 extension types
  - sbit, sfr, and sfr16
    - Declare outside of main() program
    - Essentially a friendly EQU for an SFR or SFR bit
  - bit
    - Declare anywhere a normal variable can be declared

Data type example

```
#include <REG52.h>

sbit portbit = P1^0;

void main()
{
    signed char      a = 0;
    unsigned char    b = 0;

    signed short     c = 0;
    unsigned short   d = 0;

    signed int       e = 0;
    unsigned int     f = 0;

    signed long      g = 0;
    unsigned long    h = 0;

    float            i = 0.0;
    double           j = 0.0;

    bit              k = 0;

    k = 1;

    unsigned char    l = 0;
    bit              m = 0;
}
```

Compile-time error, must declare variables at start of {}-block

#include <REG52.h>

- #include inserts text from another file

```
/*--------------------------------------------------------------------------
REG52.H

Header file for generic 80C52 and 80C32 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
--------------------------------------------------------------------------*/

#ifndef __REG52_H__
#define __REG52_H__

/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
...
```

Comments and literals

- Keil accepts C or C++ style comments:

```
// this line will be ignored by the compiler

/* these lines will
   be ignored by the compiler */


unsigned char i;   // this is ignored
unsigned char j;   /* so is this */
```

- C format for decimal/hex/octal:

```
unsigned char i = 100;    // 100 as a base 10 literal
unsigned char j = 0x64;   // 100 in hex, indicated by leading 0x
unsigned char k = 0144;   // 100 in octal, indicated by the leading 0
```

Blinking LEDs example

- Goal: Make LEDs blink every second
- Attempt 1: Big do-nothing loop

```
#include <REG52.h>

void main()
{
   unsigned int i = 0;
   unsigned char lights = 0xFF;
   P0 = lights;
   while (1)
   {
      for (i = 0; i < 30000; i++)
         ;
      lights = ~lights;
      P0 = lights;
   }
}
```

Creating a delay function

```
#include <REG52.h>

void delay(const unsigned int ms)
{
   unsigned int x;
   for (x = 0; x < ms; x++)
   {
      unsigned int y;
      for (y = 0; y <= 113; y++)
         ;
   }
}

void main()
{
   unsigned char lights = 0xFF;
   P0 = lights;
   while (1)
   {
      delay(1000);
      lights = ~lights;
      P0 = lights;
   }
}
```

We promise not to change function parameter

Found by trial and error

"Super Loop" architecture

Function order

- Function order matters in C

```
#include <REG52.h>

void main()
{
   unsigned char lights = 0xFF;
   P0 = lights;
   while (1)
   {
      delay(1000);
      lights = ~lights;
      P0 = lights;
   }
}

void delay(const unsigned int ms)
{
   unsigned int x;
   unsigned int y;

   for (x = 0; x < ms; x++)
   {
      for (y = 0; y <= 113; y++)
         ;
   }
}
```

```
Build target 'Target 1'
compiling DelayFunction.c...
DelayFunction.c(8): warning C206: 'delay': missing function-prototype
DelayFunction.c(8): error C267: 'delay': requires ANSI-style prototype
DelayFunction.c(14): error C231: 'delay': redefinition
DelayFunction.c(23): error C231: 'delay': redefinitionTarget not created
```

Function prototypes

- Declare you will later implement a function

```
#include <REG52.h>

void delay(const unsigned int ms);

void main()
{
   unsigned char lights = 0xFF;
   P0 = lights;
   while (1)
   {
      delay(1000);
      lights = ~lights;
      P0 = lights;
   }
}
void delay(const unsigned int ms)
{
   unsigned int x;
   unsigned int y;

   for (x = 0; x < ms; x++)
   {
      for (y = 0; y <= 113; y++)
         ;
   }
}
```

Parameters & return values

- Functions can:
  - Take parameters as input, empty ()'s if none
  - Return a value as output (or void if none)
  - In C, Compiler automatically maps to registers and/or memory addresses
- Example:
  - Goal: toggle LEDs if any button pushed
    - Create function to counting # of currently down buttons

Button counter

```
sbit BUTTON0 = P2^0;
sbit BUTTON1 = P2^1;
sbit BUTTON2 = P2^2;
sbit BUTTON3 = P2^3;

// Return number of buttons currently pushed
unsigned char getNumButtons()
{
    unsigned char result = 0;
    if (!BUTTON0)
        result++;
    if (!BUTTON1)
        result++;
    if (!BUTTON2)
        result++;
    if (!BUTTON3)
       result++;
    return result;
}
void main()
{
    unsigned char lights = 0xFF;
    P0 = lights;
    while (1)
   {
      if (getNumButtons() > 0)
      {
         while (getNumButtons() != 0)
            ;
         lights = ~lights;
         P0 = lights;
      }
   }
}
```

Interrupt functions

- Interrupt service routines (ISRs)
  - Special extension to denote ISR function
  - Keyword "interrupt" and a type number

Interrupt number	Description
0	External0
1	Timer0
2	External1
3	Timer1
4	Serial port
5	Timer2

Square wave using ISR

- Goal: Square wave on P1.0 using interrupts

```
// Use Timer0 and interrupt to create a 1kHz square wave on P1.0
#include <REG52.H>

sbit portbit = P1^0;

void main()
{
   TMOD = 0x02;                    // Set timer0 to auto-reload mode
   TH0  = -50;                     // 50 microseconds between overflow
   TL0  = -50;                     // Start timer at beginning value
   TR0  = 1;                       // Start timer0
   IE   = 0x82;                    // Enable timer0 interrupt
   while (1)                       // Loop forver
      ;
}

void timer0ISR() interrupt 1
{
    portbit = !portbit;            // Toggle port bit
}
```

Summary

- Introduction to C
  - 8051 specific extensions
  - Declaring variables
  - Creating functions
  - Creating ISRs