NMK322 - Microcontroller

Lecture 03 – C for 8051

Coding for 8051

- Assembly language
  - ‘human-friendly’ machine code → mnemonics
  - highest performance achievable
  - can be intimidating for beginners
- C language
  - high-level - more familiar for many
  - portable - easier to move to/from other platform
    - platform-specific syntax not portable!
  - code-size may not be optimize (speed/storage)

C for 8051

- mainly has the same syntax
  - 8051-specific usually custom keyword/syntax
- basic data types defaults to 8051-friendly size
  - int is 16-bits (2 bytes)
  - 8-bit controller → prioritize using 8-bit data!
  - 32-bit integer and real numbers can be used
  - bit data needs custom syntax
- port access = data access
  - ports are registers → similar syntax to 8-bit data
  - declaration need special syntax

8051 C Compiler

- Keil C51
  - first C compiler designed specific for 8051
  - commercial product by Keil
    - evaluation version limits code size to 2K
  - Keil (company) bought by ARM @2005
- SDCC
  - open-source compiler for various small controller
    - the only one for 8051
  - cross-platform (Linux, MacOS, Windows)
  - different syntax from Keil C51

Code51: Base Structure

- main function
  - no args
  - no return value
- code flow
  - repetitive
  - never stops
- comments
  - after //
  - between /* and */

```
void main(void) {
  // setup @ init
  while (1) {
    // task loop
  }
}
```

Code51: Declare SFR / SBIT

- Special Function Registers (SFR)
  - special system registers
  - ports are SFRs!
- SFR Bits (SBIT)
  - some SFRs are bit-addressable
  - ports are!
- Access like normal variable!

```
// Keil syntax
sfr P1 = 0x90;
sbit P2_1 = 0xA1;
sbit P1_4 = P1^4;
```

```
// sdcc syntax
__sfr __at (0x90) P1;
__sbit __at (0xA1) P2_1;
__sbit __at (0x94) P1_4;
```

Code51: Hello, Blink!

- Blinking LEDs on Port 3
  - require delay → wait counter!
  - needs large count!

```
sfr LED8 = 0xB0;
void main(void) {
  unsigned int wait;
  while (1) {
    LED8 = 0xff;
    for (wait=0;wait<50000;wait++);
    LED8 = 0x00;
    for (wait=0;wait<50000;wait++);
  }
}
```

Code51 Task: Binary counter on Port 1

Code51 Task: 7Segment 0-9 Counter on P1

Header51: reg51.h

- no need to memorize registers addresses
  - use reg51.h header file
  - all basic 8051 sfr declared
- not supported in sdcc
  - use 8051.h instead

Code51: Hello, Blink! (redo)

```
#include <reg51.h>
#define LED8 P3
void main(void) {
  unsigned int wait;
  while (1) {
    LED8 = 0xff;
    for (wait=0;wait<50000;wait++);
    LED8 = 0x00;
    for (wait=0;wait<50000;wait++);
  }
}
```

Code51 Task: Copy input as P1 and send to P2

Data51: signed 8-bit integer

- technically, that is `char`
  - some compiler refuse to treat as integer!
- requires explicit `signed` modifier
  - `signed char sign8b;`
  - accepted for integer operations, etc.

Code51 Task: Send binary values (-5 , … , 5) to P1

(Using array, `unsigned char` counter)

Code51 Task: Send binary values (-5 , … , 5) to P1
(Use counter value)

Code51 Task: Blink LED at P3.7 for 23000 times and stop

Scratch-pad: Bit addressable register

- declared using `bit` keyword
  - no need to specify address
  - handy for user flags

```
// Keil syntax
bit state;
```

```
// sdcc syntax
__bit state;
```

Code51: ‘Done’ Flag

```
#include <reg51.h>
void main(void) {
  bit done;
  while (1) {
    done = 0;
    while (done==0) {
      if (P1==0xAA) done = 1;
      // do something else here?
    }
  }
}
```

Boolean vs Bitwise Operators

- Boolean
  - AND (&&), OR (||) and INVERT (!)
  - condition for code flow
- Bitwise
  - AND (&), OR (|), XOR (^) and INVERT (~)
  - bit slicing data bytes for processing

Code51 Task: Continuously toggle alternating bits on P2

Code51 Task: Blink LED at P2.7 (only that pin!) without using sbit!

ASCII Codes

- ‘original’ character encoding
  - only 7-bits!
  - extended 8-bit version added later (non-standard?)
- nice info: alphabets
  - ‘A’ – ‘Z’ → 0x41 – 0x5B
  - ‘a’ – ‘z’ → 0x61 – 0x7B
  - {upper,lower}-case differs by 0x20
- nice info: numbers
  - ‘0’ – ‘9’ → 0x30 – 0x39
  - 0x30 offset

Code51 Task: Read 2-bit value from {P1.1,P1.0} and send an ASCII char to P2 based on given table

P1.1	P1.0	ASCII Char
0	0	‘a’
0	1	‘b’
1	0	‘c’
1	1	‘d’

**Hint: Can be done without branch/selection*

Code51: Packed BCD ↔ ASCII

```
#include <reg51.h>
#define PACKED_BCD 0x29
void main(void) {
  unsigned char temp1, temp2;
  while (1) {
    temp1 = PACKED_BCD & 0x0F;
    P1 = temp1 | 0x30; // temp1 + 0x30
    temp2 = (PACKED_BCD & 0xF0)>>4;
    P2 = temp2 | 0x30;
  }
}
```

Code51 Task: Read BCD data from P0
and send ASCII code to P1 (MSB) and P2 (LSB)

Code51 Task: Read binary from P3
and send BCD value to {P0,P1,P2}

Data Serialization

- many peripherals uses serial data
- standard ones covered
  - UART/RS232!
- some standards require software library
  - I2C , SPI
- some just create custom protocol
  - temperature sensor AM2302

Code51 Task: Read 8-bit data from P2
and send it serially on P1.3 (LSB first)

Code51 Task: Read 8-bit serial data (LSB first) from P1.2 and send it to P2

End of Lecture03