AN001

Using SDCC with Z80 Trainer Kit

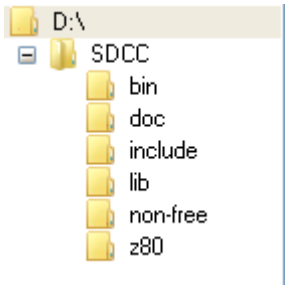
Wichit Sirichote, kswichit@kmitl.ac.th

Introduction

This applications note introduces how to use the sdcc c compiler for the Z80 Trainer Kit. The process will be startup code modification, compiling method, how to load the hex file and test run the code.

Get the compiler

The sdcc c compiler is available for download at **sdcc**.sourceforge.net. I suggest to install at drive D:\sdcc for later backup. The folder tree will look like this.



Startup code modification

The startup code is a portion of code that CPU runs after power up. The start address for Z80 is 0x0000 followed with interrupt vector locations. The startup code will set stack memory, initialize global variables then calls the main function. The source of generic startup code, ctr0.s is located at d:\sdcc\lib\source\z80.

```
;; Generic crt0.s for a Z80
      .globl      _main

        .area _HEADER (ABS)
      ;; Reset vector
      .org  0
      jp    init

      .org  0x08
      reti
      .org  0x10
      reti
      .org  0x18
      reti
      .org  0x20
      reti
      .org  0x28
      reti
      .org  0x30
      reti
      .org  0x38
      reti

      .org  0x100
init:
      ;; Stack at the top of memory.
      ld    sp,#0xffff

        ;; Initialise global variables
      call  gsinit
      call  _main
      jp    _exit
```

The bold instructions can be modified for our kit. We may remove the first block of code by adding semicolon(;) and set the start from 0x100 to 0x1800, says. The SP can be set to the new stack at 0x8FFF.

```
      .org  0x1800
  init:
      ;; Stack at the top of memory.
      ld    sp,#0x8fff
```

Then save file with the new name, mycrt0.s. Next step is to use Z80 assembler to convert it to object file. Under command prompt, set path for sdasz80 assembler.

```
D:\sdcc\lib\src\z80\path \sdcc\bin
```

Now type command to complie startup code.

```
D:\sdcc\lib\src\z80\sdasz80 -o
mycrt0.s
```

The object file will be `mycrt0.rel`, copy it to our folder d:\sdcc\z80 for linking with our c program.

LED.c test program

We can then edit the sample test code by toggle the 8-bit LED having I/O address at 0x40. SDCC uses keyword __at for locating the absolute Z80 I/O address.

```
// test sdcc for z80 trainer

__sfr __at 0x40 GPIO1;


void delay(int j)
{
      int i;
      for(i=0; i<j; i++)
      continue;

}


main()
{
      int n=0;

      while(1)
      {
        GPIO1=n++;
       delay(500);
      }

}
```

The program has only two functions i.e., delay() and main(). Delay() function is a simple for-loop counting. And main() function is forever loop writing the value of variable n (which is incremented when it was repeated) to the GPIO1.

Compiling c program

Compiling the source code is done by typing command line as follows.

```
D:\sdcc\z80\sdcc -mz80 --code-
loc 0x1900 --data-loc 0x8000
--no-std-crt0 mycrt0.rel led.c
```

The options are,

`-mz80` generate Z80 assembly code,

`--code-loc 0x1900` place the begin of code at 0x1900,

`--data-loc 0x8000` begin address of variables in RAM,

`--no-std-crt0` do not use standard startup code,

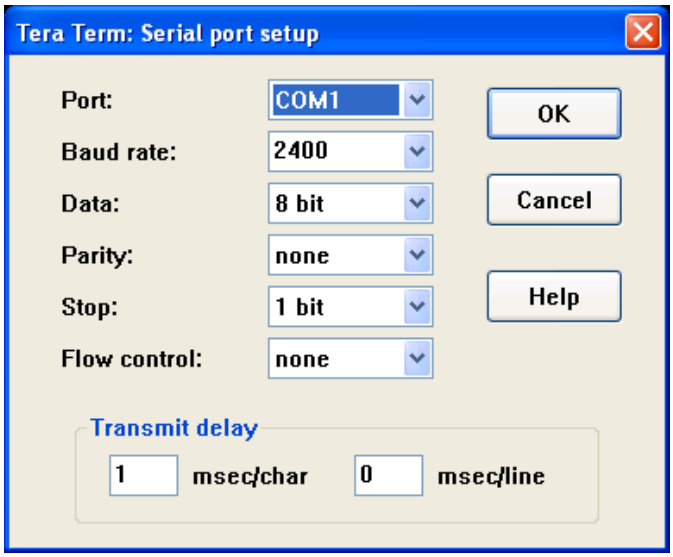
`mycrt0.rel` use this startup code instead.

`Led.c` is the source code.

Hex file downloading

If no error, the compiler will produce standard Intel hex file. For our source code led.c, the hex file will be led.ihx. This hex file can be downloaded to the trainer kit easily with RS232 cross cable. We can use PC running terminal emulator software to be hex file transmitter at speed 2400 bit/s. The example shows how to use Teraterm as the terminal emulator.

Open Serial port setup, set speed to 2400 8-data bit, no parity one stop bit and no flow control. Set transmit delay to1ms/char.



Press key Download and key GO, the Z80 trainer will wait for data stream from PC.

Click file>Send File>select the led.ihx, then ENTER

When the hex file has been received correctly, the start message will be displayed. If there is error with checksum byte, the display will show -Err.

Test code running

Simply press PC key to set current address to be executed at 1800, (the address being set for code space in the startup code). Then press GO. Did you see the binary number counting up?

More fun with LED rotation

Suppose we want the dot LED to be rotated from right to left, how the code will look like?

```
main()
{
int i=0;
int j;
      while(1)
      {
         n=1;
       for(j=0; j<8; j++)
        {
         GPIO1=n;
         n<<=1;
         delay(500);
        }
}
```

Instead of incrementing the variable n, we rotate it with c operator <<.

Try recompile the new source code and load it to the trainer board. It's fun! with c programming.

Now with left and right running.

```
main()
{
    int i=0;

while(1)
   {
    n=1;
        for(j=0; j<8; j++)
       {
          GPIO1=n;
          n<<=1;
          delay(500);
       }
       n=0x80;
        for(j=0; j<8; j++)
       {
          GPIO1=n;
          n>>=1;
          delay(500);
       }
    }
}
```

Conclusion

Using sdcc for Z80 trainer kit is simply done by modifying the startup code, crt0.s mainly at the start location of code and data memory. Compiling the c source code is done by command line with options. The sdcc will produce standard Intel hex file for directly downloading to the Z80 trainer using PC COM port.

Tools address

1. sdcc c compiler, **sdcc.sourceforge.net**
2. Terminal emulator program, **http://hp.vector.co.jp/authors/VA002416/teraterm.html**