University of British Columbia
Electrical and Computer Engineering
Electrical and Computer Engineering Design Studio
ELEC291/ELEC292

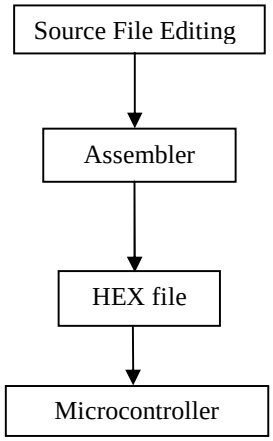
Debugging Assembly Programs with Deb51

Introduction

This document describes how to use the 8051 debugger Deb51. Deb51 is a communication interpreter that runs as a console application in MS Windows. It communicates with the target microcontroller using a serial port. In order to use Deb51, the target microcontroller must be loaded not only with the program to be debugged, but also with the debugger itself: "Debug9xx.ihx".

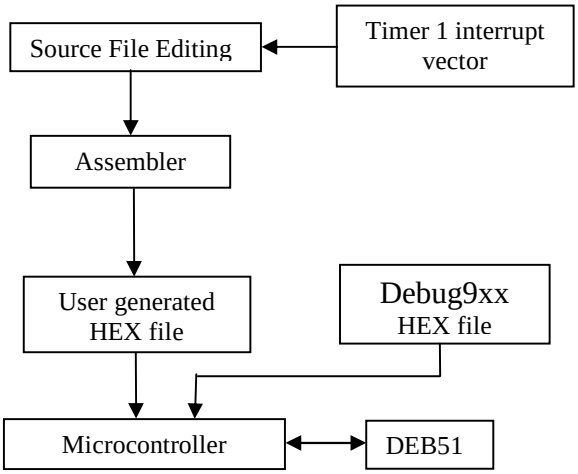
Setting up DEB51

The figure below shows the normal process of creating and flashing a program for a 8051 microcontroller. It includes compilation of the code and flashing of the microcontroller target using a suitable Integrated Development Environment such as CrossIDE.



In order to use Deb51 to debug the source code, the following steps must be added to the figure above:

1) Add an interrupt vector for timer 1 to the source code.
2) Flash the compiled source code AND the debugger hex file to the target microcontroller.
3) Run the debugger command interpreter Deb51 to communicate with the target microcontroller debugger.



Target Debugger Program Requirements

The target debugger program (Debugger_AT89LP52.hex) will need exclusive use to the following microcontroller resources in order to operate properly:

1) 64 bytes of IDATA memory starting at address C0H.
2) Timer 1 interrupt, timer 1, and the serial port.
3) 1.5k bytes of code memory starting at address 1800H.

If the user code attempts to use any of these resources, the debugger may not operate properly.

Debugging Session Example

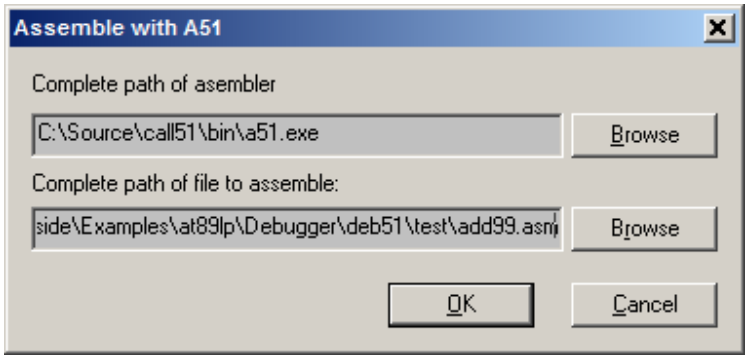
To demonstrate the use of DEB51, let us start with the small 8051 assembly program shown below. The program is supposed to add 99H to the accumulator and save the result into register B. The target microcontroller is the AT89LP52 manufactured by Atmel. We will be using CrossIDE to edit, compile, and flash the microcontroller.

```
$MODLP52

org 0000H
    ljmp myprogram

myprogram:
    mov SP, #7FH

    mov a, #10H
    add a, 99H
    mov b, a

forever:
    sjmp forever
END
```

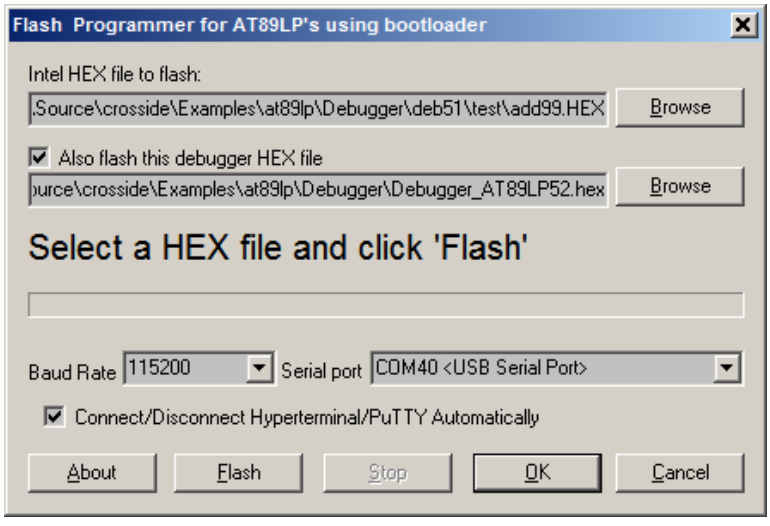
The first step towards using the debugger is adding the interrupt vector for timer 1 to the source code above:

```
$MODLP52

org 0000H
     ljmp myprogram

org 001BH
     ljmp 181BH

myprogram:
     mov SP, #7FH

     mov a, #10H
     add a, 99H
     mov b, a

forever:
     sjmp forever
END
```

The code above is assembled as usual, for example using A51:



After assembling, the corresponding “add99.HEX” file is generated. Now we ‘flash’ that file along with the debugger “Debugger_AT89LP52.hex” into the microcontroller:



To start the debugging session, run the program “deb51.exe” by clicking the icon:

After double-clicking in the icon above a console window pops open and the computer automatically scans for a microcontroller running the target debugger:

At this point we can start debugging our code. For example, let us run the “Single Step” instruction and see what happens when each line of the program is executed:

```
AT89LP52> s
PC=001e A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=07 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
001e: 75 81 7f  mov     SP,#7f
AT89LP52> s
PC=0021 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0021: 74 10     mov     a,#10
AT89LP52> s
PC=0023 A=10 PSW=01 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0023: 25 99     add     a,SBUF
AT89LP52> s
PC=0025 A=55 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0025: f5 f0     mov     B,a
AT89LP52> s
PC=0027 A=55 PSW=00 B=55 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0027: 80 fe     sjmp    0027
AT89LP52> s
PC=0027 A=55 PSW=00 B=55 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0027: 80 fe     sjmp    0027
AT89LP52>
```

Notice from the step by step execution above that the "**`add a, SBUF`**" instruction "was not" in our code. Actually we made a mistake when writing the code as we forgot to put '#" before the add instruction. Therefore the assembler assumed that we were using SFR 99H (whose name is SBUF) instead of number 99H. The corrected source code is shown below:

```
$MODLP52

org 0000H
    ljmp myprogram

org 001BH
    ljmp 1903H

myprogram:
    mov SP, #7FH

    mov a, #10H
    add a, #99H
    mov b, a

forever:
    sjmp forever
END
```

When the fixed code is run step by step using the debugger (following the steps described above) the program behaves as expected:

```
AT89LP52> s
PC=001e A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=07 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
001e: 75 81 7f  mov     SP,#7f
AT89LP52> s
PC=0021 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0021: 74 10     mov     a,#10
AT89LP52> s
PC=0023 A=10 PSW=01 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0023: 24 99     add     a,#99
AT89LP52> s
PC=0025 A=a9 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0025: f5 f0     mov     B,a
AT89LP52> s
PC=0027 A=a9 PSW=00 B=a9 IE=08 DPL=00 DPH=00 SP=7f REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=fa R5=e9 R6=fc R7=ef
0027: 80 fe     sjmp    0027
AT89LP52>
```

DEB51 Command Summary

Single Step

Syntax:

s [n]
step [n]

Description:

Execute [n] assembly instructions starting from the current program counter. After the last assembly instruction the registers are printed.

Examples:

Step one assembly instruction:

```
AT89LP52> s
PC=0023 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=07 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
0023: 75 81 3e  mov     SP,#3e
```

Step three assembly instructions:

```
AT89LP52> s 3
0026: 12 05 c2  lcall   05c2
05c2: 75 82 00  mov     DPL,#00

PC=05c5 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=40 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
05c5: 22        ret
```

Trace Step

Syntax:

t [n]
trace [n]

Description:

Execute [n] assembly instructions starting from the current program counter. After each assembly instruction the registers are printed.

Examples:

Trace one assembly instruction:

```
AT89LP52> t
PC=0023 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=07 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
0023: 75 81 3e  mov     SP,#3e
```

Trace three assembly instructions:

```
AT89LP52> t 3
PC=0026 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=3e REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
0026: 12 05 c2  lcall   05c2

PC=05c2 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=40 REGBANK:0
```

```
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
05c2: 75 82 00  mov     DPL,#00

PC=05c5 A=00 PSW=00 B=00 IE=08 DPL=00 DPH=00 SP=40 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
05c5: 22        ret
```

Go

Syntax:

```
g
go
run
```

Description:

Execute the microcontroller code at full speed.

Example:

```
AT89LP52> g
Switching to terminal mode.  Press F1 for help.
```

Notes:

If the running program uses the serial port for I/O, the data can be read/displayed in deb51. The terminal emulator is configured at 115200 baud, no parity, 8 data bits, and one stop bit. To return to command mode press key F10.

Display Registers

Syntax:

```
r
```

Description:

Display the microcontroller's main registers and the current assembly instruction as pointed by the PC.

Example:

```
AT89LP52> r
PC=0000 A=00 PSW=00 B=00 IE=00 DPL=00 DPH=00 SP=07 REGBANK:0
R0=c3 R1=7b R2=7e R3=e7 R4=b6 R5=e9 R6=fc R7=eb
0000: 02 00 23  ljmp    0023
```

Dis-assemble

Syntax:

```
u [address] [number]
```

Description:

Disassemble [number] opcodes starting at [address].

Example:

```
AT89LP52> u ff00 10
ff00: 75 a2 08  mov     AUXR1,#08
ff03: 90 ff 08  mov     dptr,#ff08
ff06: 23        rl      a
ff07: 73        jmp     @a+dptr
ff08: e1 4c     ajmp    ff4c
ff0a: e1 9e     ajmp    ff9e
ff0c: e1 21     ajmp    ff21
ff0e: e1 18     ajmp    ff18
ff10: e1 37     ajmp    ff37
ff12: e1 84     ajmp    ff84
ff14: e1 8b     ajmp    ff8b
ff16: e1 44     ajmp    ff44
ff18: 8f e6     mov     FMADRL,r7
ff1a: 75 e4 6c  mov     FMCON,#6c
ff1d: af e5     mov     r7,FMDATA
ff1f: e1 a3     ajmp    ffa3
```

Display data memory

Syntax:

d [address] [number]

Description:

Display [number] of internal data bytes starting at [address].

Example:

```
AT89LP52> d 20 30
D:20:  bb fb bb 52 2c 00 da 59 : 34 86 7a ff b5 82 df 9f  ...R,..Y4.z.....
D:30:  32 f7 cf c7 ea ff 37 0f : dc 41 e6 1b af e5 0f 29  2.....7..A.....)
D:40:  00 c5 05 00 00 00 00 df : 67 90 b6 fe fb 57 cd 8b  ........g....W..
```

Display Expanded Memory

Syntax:

x [address] [number]

NOTE**: Unfortunately the AT89LP52 has no expanded memory.**

Description:

Display [number] of external data bytes starting at [address].

Example:

```
AT89LP52> x 0 40
X:0000:  d4 6e ee cf ee eb b2 ca : 7f ff 7f 12 9a ce b3 7d  .n.............}
X:0010:  fe e9 33 1d 7d b2 af d5 : 7f 9b 7e 16 ef 3d 7e ff  ..3.}.....=..
X:0020:  ae 3d f8 5b ff 6f 62 cd : 91 fa a3 55 5a 72 aa da  .=.[.ob....UZr..
X:0030:  f0 ff c0 f3 7f de 0d 2b : f5 85 fe ef f3 2d 2f 22  .......+.....-/"
```

Display Code Memory

Syntax:

c [address] [number]

Description:

Display [number] of code data bytes starting at [address].

Example:

```
AT89LP52> c 1900 20
C:1900:  21 ea 00 21 0b b1 65 80 : fe 21 05 c2 af c0 d0 c0  !..!..e..!......
C:1910:  83 c0 82 c0 e0 90 01 00 : 75 d0 00 e8 f0 78 01 a3  ........u....x..
```

Fill Data Memory

Syntax:

fd [address] [number] [value]

Description:

Fill [number] of internal data bytes starting at [address] with [value].

Example:

```
AT89LP52> fd 20 10 55
AT89LP52> d 20 10
D:20:  55 55 55 55 55 55 55 55 : 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
```

Fill Expanded Memory

Syntax:

fx [address] [number] [value]

NOTE**: Unfortunately the AT89LP52 has no expanded memory.**

Description:

Fill [number] of expanded data bytes starting at [address] with [value].

Example:

```
AT89LP52> fx 0 20 0
AT89LP52> x 0 20
X:0000:  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
X:0010:  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
```

Exit

Syntax:

q
quit
exit

Description:

Exits deb51.

Example:

AT89LP52> q
Press any key to exit...

Display/Change General Purpose Registers

Syntax:

r0[=value]
r1[=value]
r2[=value]
r3[=value]
r4[=value]
r5[=value]
r6[=value]
r7[=value]

Description:

Display the selected register or changes it to [value].

Example:

```
AT89LP52> r0=01
AT89LP52> r0
01
```

Notes:

The r0 to r7 commands display/change the register from the register bank selected with bits RS0 and RS1 of the PSW register.

Restore Main Registers to Their Default Values

Syntax:

restore

Description:

Sets the main registers to their default values.

Example:

```
AT89LP52> r
PC=0056 A=44 PSW=00 B=00 IE=08 DPL=00 DPH=42 SP=3e REGBANK:0
R0=ff R1=00 R2=7e R3=e7 R4=a6 R5=e9 R6=fc R7=ef
0056: d8 fd     djnz    r0,0055
AT89LP52> restore
PC=0000 A=00 PSW=00 B=00 IE=00 DPL=00 DPH=00 SP=07 REGBANK:0
R0=ff R1=00 R2=7e R3=e7 R4=a6 R5=e9 R6=fc R7=ef
0000: 02 00 23  ljmp    0023
```

Edit Data Memory

Syntax:

ed [address]

Description:

Edits data memory starting at [address]. Pressing <Space> moves

the cursor to the next location. Pressing <Enter> finishes this command.

Example:

```
AT89LP52> d 30 10
D:30:  b2 f7 cf c7 fa ff 77 0f : d8 49 e4 1b af e5 07 56  ......w..I.....V
AT89LP52> ed 30
30:   b2.55  f7.66  cf.77  c7.88  fa.
AT89LP52> d 30 10
D:30:  55 66 77 88 fa ff 77 0f : d8 49 e4 1b af e5 07 56  Ufw...w..I.....V
```

Edit Expanded Memory

Syntax:

ex [address]
NOTE**: Unfortunately the AT89LP52 has no expanded memory.**

Description:

Edits expanded memory starting at [address]. Pressing <Space> moves the cursor to the next location. Pressing <Enter> finishes this command.

Example:

```
AT89LP52> x 80 10
X:0080:  f7 77 a3 b3 57 77 9d 2b : 77 ad d1 dd 6a 2f 7c f2  .w..Ww.+w...j/|.
AT89LP52> ex 80
0080:   f7.00  77.01  a3.02  b3.03  57.
AT89LP52> x 80 10
X:0080:  00 01 02 03 57 77 9d 2b : 77 ad d1 dd 6a 2f 7c f2  ....Ww.+w...j/|.
```

Display/Change Special Function Registers

Syntax:

ACC[=value]
A[=value]
PC[=value]
PSW[=value]
B[=value]
IE[=value]
DPL[=value]
DPH[=value]
DPTR[=value]
SP[=value]
[Any SFR name][=value]

Description:

Display the selected register or changes it to [value]. Any SFR name for the current microcontroller can be displayed or changed this way. Bits values can also be changed individualy.

Examples:

```
AT89LP52> ACC
00
AT89LP52> ACC.0=1
```

```
AT89LP52> ACC
01
AT89LP52> P0M1=0
AT89LP52> P0M2=0
AT89LP52> P0.0=1
AT89LP52> P0.0
1
```

Display/Change Special Function Bits

Syntax:

CY[=value]
AC[=value]
FO[=value]
RS1[=value]
RS0[=value]
OV[=value]
UD[=value]
P[=value]
[Any SFR bit name][=value]

Description:

Display the selected bit or changes it to [value]. Any bit name for the current microcontroller can be displayed or changed this way.

Examples:

```
AT89LP52> cy=1
AT89LP52> cy
1
AT89LP52> cy=0
AT89LP52> cy
0
```

Flush the Serial Receive Buffer

Syntax:

flush

Description:

Flushes the host serial port, so communication with the target serial port can be re-established.

Example:

```
AT89LP52> flush
```

Switch to Terminal Mode

Syntax:

term
<F10>

Description:

Switches from debug mode to terminal mode. To return to debug mode, press <F10>

Example:

```
AT89LP52> term
Switching to terminal mode.  Press F1 for help.
```

Clear the Screen

Syntax:

cls
<F2>

Description:

Clears the screen.

Example:

```
AT89LP52> cls
```

Release the Host Computer Serial Port

Syntax:

h
hangup

Description:

Temporarily releases the host computer serial port, so it can be used by another application.

Example:

```
AT89LP52> h
Serial port COM1 temporarily released. Press <Enter> to reconnect...
```