Lab1 - DVA454

Peter Andersson (pan15005@student.mdh.se)

September 12, 2018

1

The debugging session is terminated by pressing "stop debugging" in Atmel Studio.

2

The program on the dev. board is terminated by either pressing the "erase now" button in the memories option of the device programming window or by uploading a new program to it.

3

3.1

The variables used in the program is of type "unsigned char", which means that they are 8 bits and positive only. I will write a small explanation to every operator and also provide a screen shot from the debugging session. The debugging was done using the simulation tool in *Atmel Studio 7*.

1. AND-operator returns a '1' iff both bits compared are set to 1.
   op1 = 0x01 = 0000 0001
   op2 = 0x03 = 0000 0011
   result = op1 & op2 = 0000 0001

```
/* Bitwise AND */
op1 = 0x01;
op2 = 0x03;
result = op1 & op2;
```
result 1

Figure 1: AND-operator. Result = 1

2. op1 = 0x03 = 0000 0011
   op2 = 0x07 = 0000 0111
   result = op1 & op2 = 0000 0011

```
op1 = 0x03;
result = 0x07;
result &= op1;  /* result = result & op2 */
```
result 3

Figure 2: AND-operator. Result = 3

3. OR-operator returns a '1' iff any of the bits compared are set to 1.
   op1 = 0x01 = 0000 0001
   op2 = 0x03 = 0000 0011
   result = op1 | op2 = 0000 0011

```
/* Bitwise OR */
op1 = 0x01;
op2 = 0x03;
result = op1 | op2;
```
result 3

Figure 3: OR-operator. Result = 3

4. XOR-operator returns a '1' iff one of the bits compared are set to 1.
   op1 = 0x01 = 0000 0001
   op2 = 0x03 = 0000 0011
   result = op1 $\oplus$ op2 = 0000 0010

```
/* Bitwise XOR */
op1 = 0x01;
op2 = 0x03;
result = op1 ^ op2;
```
result 2

Figure 4: XOR-operator. Result = 2

5. NOT-operator returns a '1' iff a bit is set to 0, else it returns a 0.
   op1 = 0x01 = 0000 0001
   result = ¬ op1 = 1111 1110

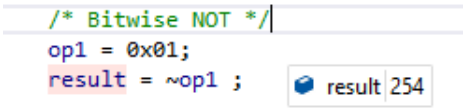

Figure 5: NOT-operator. Result = 254

6. Left-shift-operator moves the bits to the left of the operator, as many steps as the number to the right of the operator says, to the right.
   op1 = 0x02 = 0000 0010
   result = (op1 ≫ 1) = 0000 0001

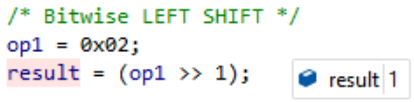

Figure 6: Left-shift-operator. Result = 1

7. op1 = 0x80 = 1000 0000
   result = (op1 ≫ 4) = 0000 1000

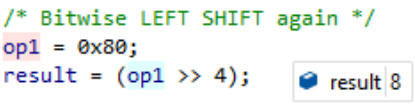

Figure 7: Left-shift-operator. Result = 8

8. Right-shift-operator moves the bits to the left of the operator, as many steps as the number to the right of the operator says, to the left.
   op1 = 0x01 = 0000 0001
   result = (op1 $\ll$ 1) = 0000 0010

```
/* Bitwise RIGHT SHIFT */
op1 = 0x01;
result = (op1 << 1);
```
result 2

Figure 8: Right-shift-operator. Result = 2

9. op1 = 0x01 = 0000 0001
   result = (op1 $\ll$ 7) = 1000 0000

```
/* Bitwise RIGHT SHIFT again */
op1 = 0x01;
result = (op1 << 7);
```
result 128

Figure 9: Right-shift-operator. Result = 128

10. The logical comparison && in an if-statement is true iff both sides are positive.
    op1 = 0x01 = 0000 0001 ¿ 0
    op2 = 0x03 = 0000 0011 ¿ 0

```
/* Logical Comparision AND */
op1 = 0x01;
op2 = 0x03;
if(op1 && op2) {
    result = TRUE; /* true */
}
else {
    result = FALSE; /* not true */
}
```
result 1

Figure 10: If-statement is true.

11. The logical comparison || in an if-statement is true iff either side is positive.
    op1 = 0x01 = 0000 0001 ¿ 0
    op2 = 0x03 = 0000 0011 ¿ 0

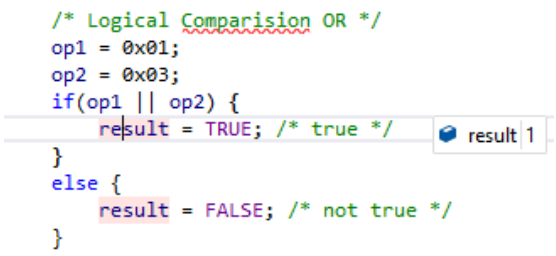

Figure 11: If-statement is true. Result = 1

3.2

At the end of the program there is an infinite loop. This because in embedded systems, normally the program should never terminate.