Common Coding Pitfalls

There are many Coding Pitfalls, each coder has its own (as well as its own style). This Page is meant as a Collection about many Errors, feel free to add your own.

Please note that we have separate Articles about various Initialisation Issues.

The DOKE-Dilemma

To put an “immediate” 16bit value into two adjacent Memory-Addresses is a common task. We 6502 Coders agree to store all 16bit values low-byte-first, msb-byte-last order.

but instead of

```
ldx#<value
ldy#>value
stx address
sty address+1
```

i prefer

```
lda#<value
sta address
lda#>value
sta address+1
```

The main reason is that the xy-method can be easily typed in wrongly, by confusing xy at the store commands, or even sta instead of stx because sta is used more often then stx or sty.

And even if you use a macro, you might be safer by not thrashing x and y as a side effect.

```
#BEGINDEF doke(_address,_value)
lda#<_value
sta _address
lda#>_value
sta _address+1
#ENDDEF
```

The above Code is inside util.mac for the k2asm.

To sum it up: When doing a doke, several spelling Mistakes could occur (e.g. 2times <, or forgetting the +1). If you use a macro, the accu-only version might be better.

Selfmodifying Madness

The 6502 has always been a popular target for self-modifying code. It doesn't have a cache, and in many Situations, self-modifying code is faster then indirect Addressing modes. Even Commodore used self-modifying Code to read Basic-Tokens.

Most of the time, the address of an absolute opcode gets modified.

```
lda#255
loop:
smod=*+1
 sta $0400
 inc smod
 bne loop
 rts
```

There are some things to take care of: You should use a coherent Style. In the above Example, you could use loop+1 instead of smod, but if you forget the +1, you modify the opcode, which would crash your program very soon.

If you use the smod=*+1 style, you can use smod just like a normal 16bit pointer.

Mixing up addressing modes

Another common error is to write “cmp 4” instead of “cmp #4”, or “lda 5” instead of “lda #5”. The assembler will not complain and the syntax highlighting in your editor may not give any indications of spelling mistakes in the code, but there is of course all the difference in the world between them…

Program counter after label

You might be tempted to use a label as a spacer between pieces of code:

```
     * = $1000
     LDA #<IRQ
     STA $0314
     LDA #>IRQ
     STA $0315
     RTS
IRQ
     INC $d020
     JMP $ea31
```

Now, if you'd want your IRQ routine at a certain location, the wrong way to do this is:

```
     * = $1000
     LDA #<IRQ
     STA $0314
     LDA #>IRQ
     STA $0315
     RTS
```

```
IRQ   * = $1100
      INC $d020
      JMP $ea31
```

The new program counter is now BEHIND the label, so IRQ points to the byte right after RTS ($100b) instead of the location wheren INC is (in this case that would be $1100). The right way to do it is:

```
      * = $1000
      LDA #<IRQ
      STA $0314
      LDA #>IRQ
      STA $0315
      RTS
      * = $1100
IRQ
      INC $d020
      JMP $ea31
```