Lucrul cu registrele la nivel de bit

În cazul ATmega 324, registrele sunt pe 8 biți. Majoritatea biților au un anumit rol, documentat în datasheet, și pot fi folosiți în următoarele scopuri:

- **biți de control**: folosiți pentru a controla diferite moduri de operare ale microcontrollerului, pentru a activa anumite componente etc. Exemplu: bitul *RXENn* din registrul `UCSRnB` permite, atunci când este setat, primirea de date pe *USARTn* (unde n este 0 sau 1).
- **biți de stare**: folosiți pentru a verifica starea unei acțiuni. Aceștia sunt, în general, read-only. Exemplu: bitul *RXCn* din registrul *UCSRnA* este setat atunci când există date de citit în bufferul de primire date pentru *USARTn* (unde n este 0 sau 1).
- **biți de date**: folosiți pentru a oferi microcontrollerului date pentru a fi prelucrate sau pentru a prelua date de la acesta. Exemplu: grupul de 8 biți *RXBn[7:0]* sunt folosiți pentru a primi date de la *USARTn*, iar grupul *TXBn[7:0]* sunt folosiți pentru a scrie date în bufferul destinat transmisiei pe *USARTn* (unde n este 0 sau 1).
- **biți rezervați**: în prezent nu sunt folosiți de către microcontroller.

O operație pe biți este o operație care afectează unul sau mai mulți biți dintr-un câmp de biți sau din reprezentarea binară a unui număr. Deoarece aceste acțiuni sunt suportate direct de către microcontroller, ele sunt cu mult mai eficiente decât operațiile de împărțire, înmulțire și chiar decât adunare.

Operații pe bytes

Operație	Formă
Scriere bit pe 1	`Byte |= (1 << Bit_index)`
Scriere bit pe 0	`Byte &= ~(1 << Bit_index)`
Toggle bit	`Byte ^= (1 << Bit_index)`
Selectare bit	`Byte & (1 << Bit_index)`

Shiftare dreapta

Shiftarea la dreapta mută fiecare bit al operandului spre dreapta cu un număr de poziții. Operandul din dreapta reprezintă numărul de biți cu care se face shiftarea operandului din stânga. Dacă este nevoie, se adaugă zerouri la stânga numărului pentru a îi menține dimensiunea.

Exemple:

11001011 >> 2 = 00110010

11001011 >> 3 = 00011001

Shiftarea la dreapta poate fi folosită pentru a împărți un număr binar la 2 (sau puteri ale lui 2) într-o singură operație. Exemplu: `1010 >> 1 = 0101`, unde:

- 1010(baza 2) = 10(baza 10)
- 0101(baza 2) = 5(baza 10)

Shiftare stânga

Shiftarea la dreapta mută fiecare bit al operandului spre stânga cu un număr de poziții. Operandul din dreapta reprezintă numărul de biți cu care se face shiftarea operandului din stânga. Dacă este nevoie, se adaugă zerouri la dreapta numărului pentru a îi menține dimensiunea.

Exemple:

11001011 << 2 = 00101100
11001011 << 3 = 01011000

Shiftarea la stânga poate fi folosită pentru a înmulți un număr cu 2 (sau puteri ale lui 2) într-o singură operație. Exemplu: `0011 << 1 = 0110`, unde:

- 0011(baza 2) = 3(baza 10)
- 0110(baza 2) = 6(baza 10)

Măști pe biți

O mască pe biți reprezintă un câmp de biți folosit pentru a seta / șterge mai mulți biți dintr-un alt câmp de biți într-o singură operație.

- Mascare biți pe 1 (setare multiplă)
  - Exemplu: 10100 | 11100 = 11100
- Mascare biți pe 0 (ștergere multiplă)
  - Exemplu: 10101 & 11100 = 10100
- Verificare status bit
  - Exemplu: 11011 & 00010 = 00010
- Toggle valoare biți
  - Exemplu: 11010011 ^ 00001111 = 11011100

Greșeli frecvente în lucrul cu biți

- **Confundarea operatorilor ”!” și ”~”.**
  - ”!” transformă ceva “fals” (=0) în “adevărat” și vice-versa.
  - ”~” transformă toate zerourile în 1 și vice-versa.
  - Exemplu: !1100 = 0; ~1100 = 0011

- **Confundarea operatorilor ”&” și ”&&”.**
  - ”&” efectuează operația de intersecție a doi biți.
  - ”&&” returnează starea de adevăr a expresiei, evaluată cu tabelul de adevăr AND, în care 0 reprezintă “fals” și orice altceva reprezintă “adevărat”.

- Exemple:
  - 1101 & 1001 = 1001; 1101 && 1001 = 1
  - 1100 & 0001 = 0000; 1100 && 0001 = 1

- **Confundarea operatorilor "|" și "||".**
  - "|" efectuează operația de uniune a doi biți.
  - "||" returnează starea de adevăr a expresiei, evaluată cu tabelul de adevăr OR, în care 0 reprezintă "fals" și orice altceva reprezintă "adevărat".
  - Exemple:
    - 1101 | 1001 = 1101; 1101 || 1001 = 1
    - 1000 | 0001 = 1001; 1000 || 0001 = 1

- Folosirea expresiei **Byte = 1 << Bit_index** când se dorește doar setarea bitului de la indexul Bit_index din octetul Byte.
  - Corect: Byte |= 1 << Bit_index
  - De ce este greșit? Pentru că șterge toți ceilalți biți din Byte. Practic, îi atribuie lui Byte valoarea lui 1 << Bit_index, care nu este altceva decât masca pe care dorim s-o aplicăm.
  - Exemplu: inițial Byte = 11001110 și Bit_index = 4.
    - Greșit: Byte = 1 << 4; ⇒ Byte devine 00010000
    - Corect: Byte |= 1 << 4; ⇒ Byte devine 11011110

- Folosirea expresiei **Byte = ~(1 << Bit_index)** când se dorește doar ștergerea bitului de la indexul Bit_index din octetul Byte.
  - Corect: Byte &= ~(1 << Bit_index)
  - De ce este greșit? Pentru că șterge toți ceilalți biți din Byte. Practic, îi atribuie lui Byte valoarea lui ~(1 << Bit_index), care nu este altceva decât masca pe care dorim s-o aplicăm.
  - Exemplu: inițial Byte = 11001110 și Bit_index = 2.
    - Greșit: Byte = ~(1 << 2); ⇒ Byte devine 11111011
    - Corect: Byte &= ~(1 << 2); ⇒ Byte devine 11001010