Microprocesoare, Microcontrolere –Prelegere 3, partea I-a

1. ATMEGA16

Una dintre cele mai populare familii microcontrolere este ATmega de la Atmel. În această prelegere va fi prezentat un reprezentat al acestei familii, şi anume ATmega16. În schema bloc din figura 1 se identifică următoarele blocuri:

- CPU pe 8 biţi;
- 16KB memorie flash pentru program;
- 1KB octeţi de memorie RAM pentru date;
- 512 octeţi de memorie EEPROM
- 32 de linii I/O bidirecţionale adresabile individual sau în grupe de 8 (4 porturi I/O);
- 3 numărătoare programabile;
- USART full duplex;
- bloc de întreruperi cu 20 de surse de întrerupere;
- generator de ceas încorporat.

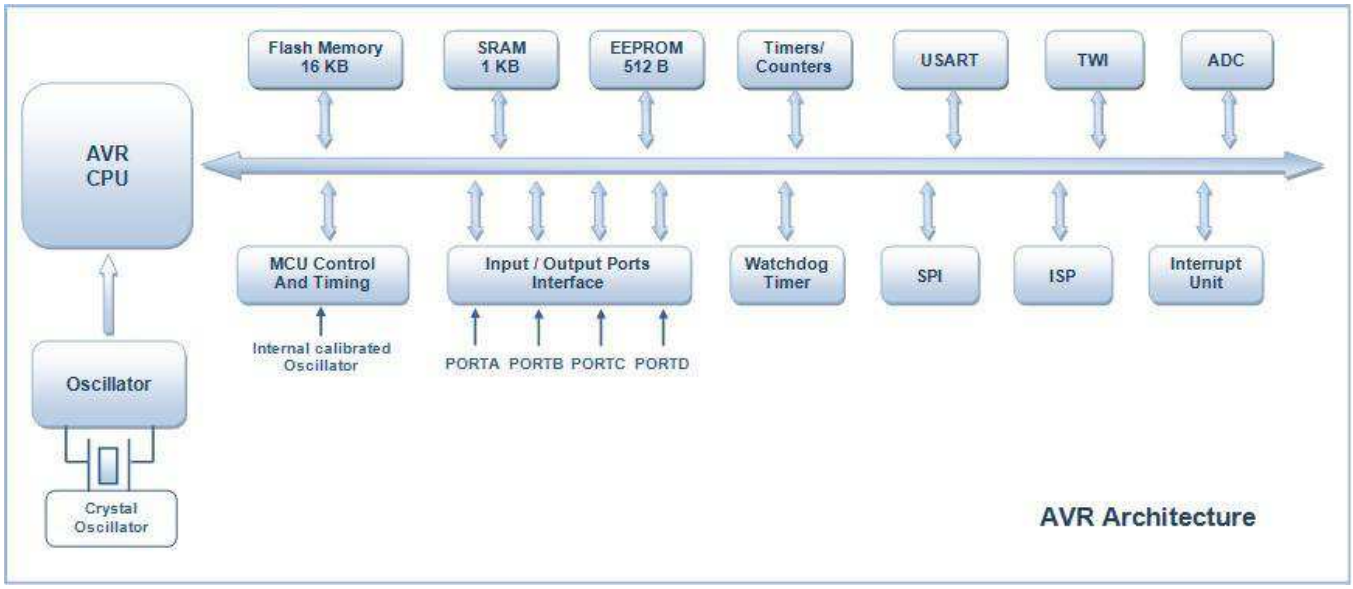


figura 1

Organizarea memoriei

Arhitectura AVR, din care face parte şi ATmega16, este o arhitectură RISC de tip Hardvard. Astfel ATmega16 este prevăzut cu spaţii de adresă separate pentru program şi date, aşa cum se prezintă în figura 2. ATmega16 este prevăzut cu 3 tipuri de memorie: de cod, de date şi EEPROM.

Memoria de program. ATmega16 conține 16KB de memorie flash pentru memorarea programului. Deoarece toate instrucţiunile AVR au o lungime de 16 sau 32 de biţi, această memorie are organizarea 8Kx16.

Memoria de date SRAM. Memoria de date este accesată la nivel de octet şi este organizată în 3 secțiuni: 32 de registre generale, 64 de registre pentru transferurile de I/O (porturi) şi 1024 de locaţii de RAM. Acest spaţiu de memorie suportă 5 moduri de adresare: direct, indirect, indirect cu deplasament, indirect cu predecrementare şi indirect cu postincrementare. Registrele R26 – R31 se cuplează două câte două şi formează registrele duble (de 16 biţi) X, Y şi Z, folosite pentru adresarea indirectă.

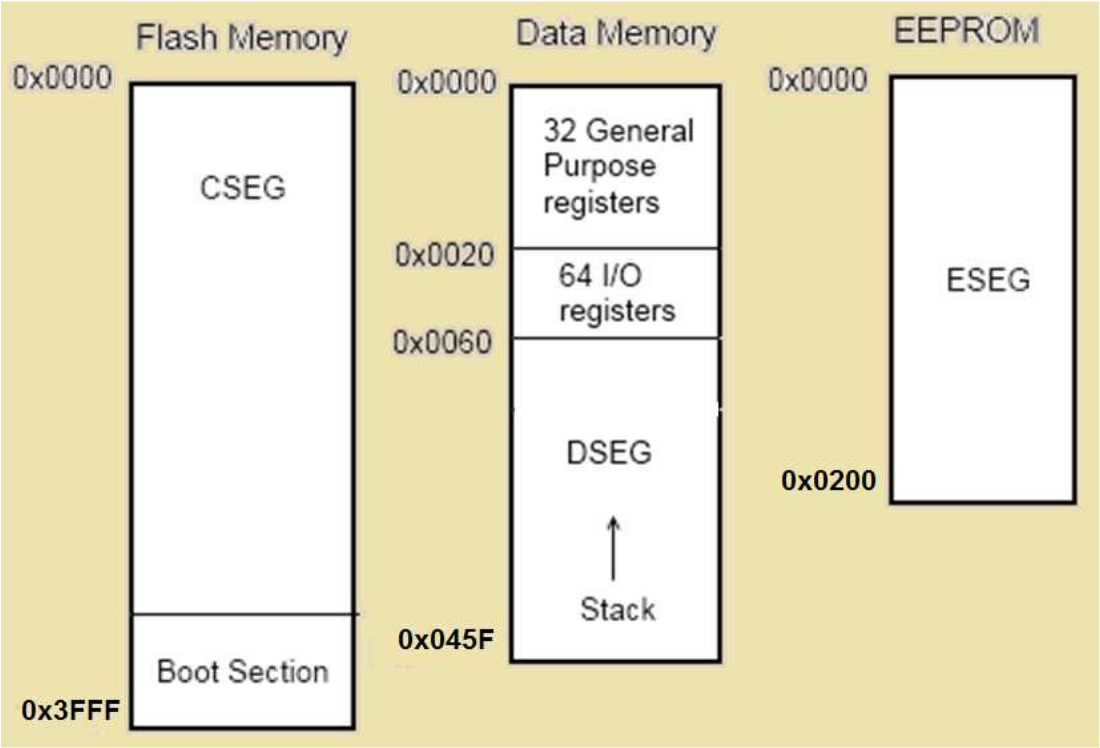


figura 2

Zona de memorie 0x0020-0x005f este alcătuită din 64 de registre I/O. În această zonă sunt mapate cele 4 porturi, registrele de control şi stare aferente celor trei timere, interfeţei seriale, stack pointer şi alte resurse ce vor fi detaliate ulterior. Structura acestei zone este prezentată în datele de catalog, la pagina 334.

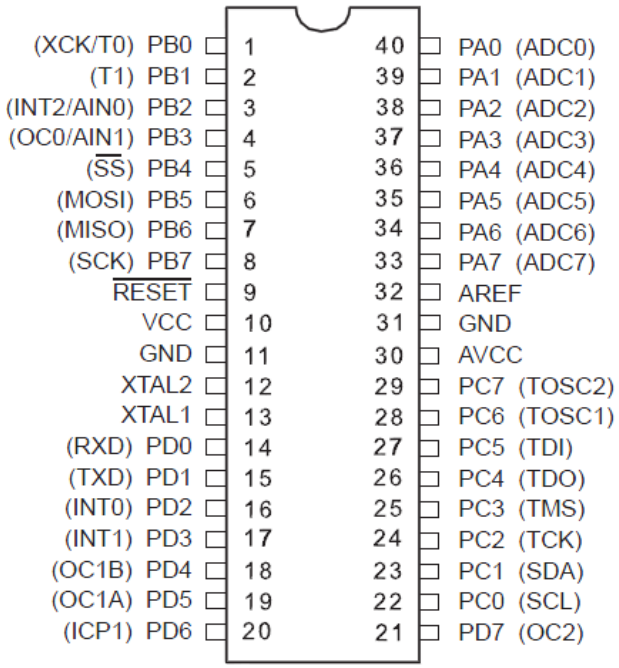
O variantă de ATmega16, şi anume ATmega162, permite adăugarea de memorie de date în exteriorul microcontrolerului. În momentul în care se plasează memorie în spaţiul extern de cod porturile A şi C se transformă în magistrale de date şi de adresă.

Memoria EEPROM. Atmeg16 conține 512 octeți de memorie EEPROM.

Setul de instrucţiuni ISA.
Treceţi în revistă modurile de adresare şi instrucţiunile ATmega16, din datele de catalog. Descrierea arhitecturii AVR este disponibilă la adresa http://www.atmel.com/images/doc0856.pdf. Deşi setul de instrucţiuni nu constituie subiect de examen, cunoaşterea lui este necesară pentru depanarea programelor.

2. Porturile A, B, C şi D.

ATmega16 comunică cu exteriorul prin intermediul a 32 de pini organizaţi în 4 grupe notate A, B, C şi D. În fiecare grupă sunt 8 pini numerotaţi de la 0 la 7. Configuraţia terminală a microcontrolerului ATmega16 este prezentată în figura următoare:



La proiectarea oricărui microcontroler trebuie luată o decizie: câte porturi să fie de porturi de ieşire şi câte porturi de intrare. La această întrebare este foarte greu de răspuns deoarece depinde de aplicaţie: există aplicaţii în cere este nevoie de multe porturi de ieşire şi puţine porturi de intrare şi aplicaţii în care este invers. Soluţia cea mai bună este să lăsăm utilizatorul să aleagă cum să anume să fie folosit un pin procesor: pentru ieşire sau pentru intrare. Soluţia de principiu este prezentată în figura următoare:

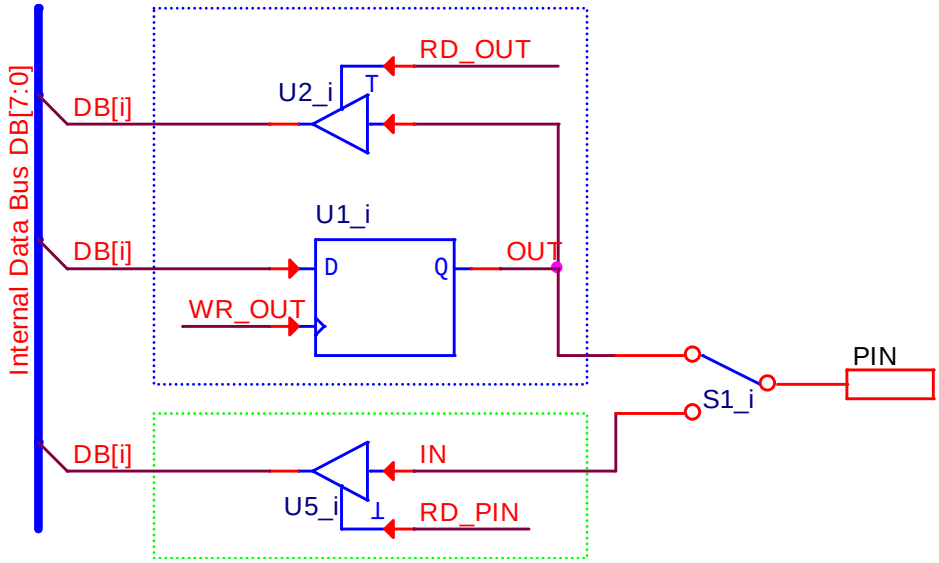


figura 3

La pinul PIN al microcontrolerului se poate conecta fie o ieşire, fie o intrare. Ieşirea este un bit dintr-un port de tip buffer şi este implementat cu o pereche bistabil-driver 3-state. Portul buffer este prezentat în prelegerea 2- fig. 1 (mai puţin LED-ul). Implementarea ieşirii este încercuită de un dreptunghi albastru din figura 3. Referinţa bistabilului este U1_i iar cea a driverului 3-state este U2_i.

Intrarea este un bit dintr-un port de intrare implementat prin driver-ul 3-stare cu referinţa U5_i şi este încercuit de dreptunghiul verde din figura 3. Portul de intrare este prezentat în prelegere 2-fig. 2, partea dreaptă.

Comutatorul S1_i decide cine anume se cuplează la pinul PIN al microcontrolerului: ieşire OUT din portul buffer sau intrarea IN din portul de intrare. Acest comutator este un comutator electronic

comandat. În figură S1_i a fost reprezentat ca un comutator mecanic pentru a-i evidenţia rolul. Semnalele WR_OUT, RD_OUT şi RD_PIN sunt generate de decodificatorul de adrese.

În continuare vom rafina decizia privitoare la numărul de porturi de intrare şi de ieşire. La ATmega16 există 32 de structuri ca cea din figura 3 grupate în 4 grupuri de 8. Pentru o mai mare flexibilitate fiecare comutator este comandat individual în loc de o comandă per grup de 8. O singură comandă per grup de 8 pini duce la doar 16 configuraţii posibile: toate cele 4 grupe sunt porturi de ieşire (OOOO), 3 porturi de ieşire şi un port de intrare (OOOI), OOIO, OOII, …, IIII. O singură comandă per grup de 8 face imposibilă o configuraţie cu 20 de ieşiri şi 12 intrări. Configurarea individuală a fiecărui pin face posibila orice configuraţie de tip *n* ieşiri şi 32-*n* intrări (n=1..32).

Comanda comutatorului se face cu un bit dintr-un al treilea port. Acest port este de tip buffer şi are binecunoscuta structură bistabil – driver 3-state. Bitul din portul de comandă este încercuit de un dreptunghi roşu în figura următoare:

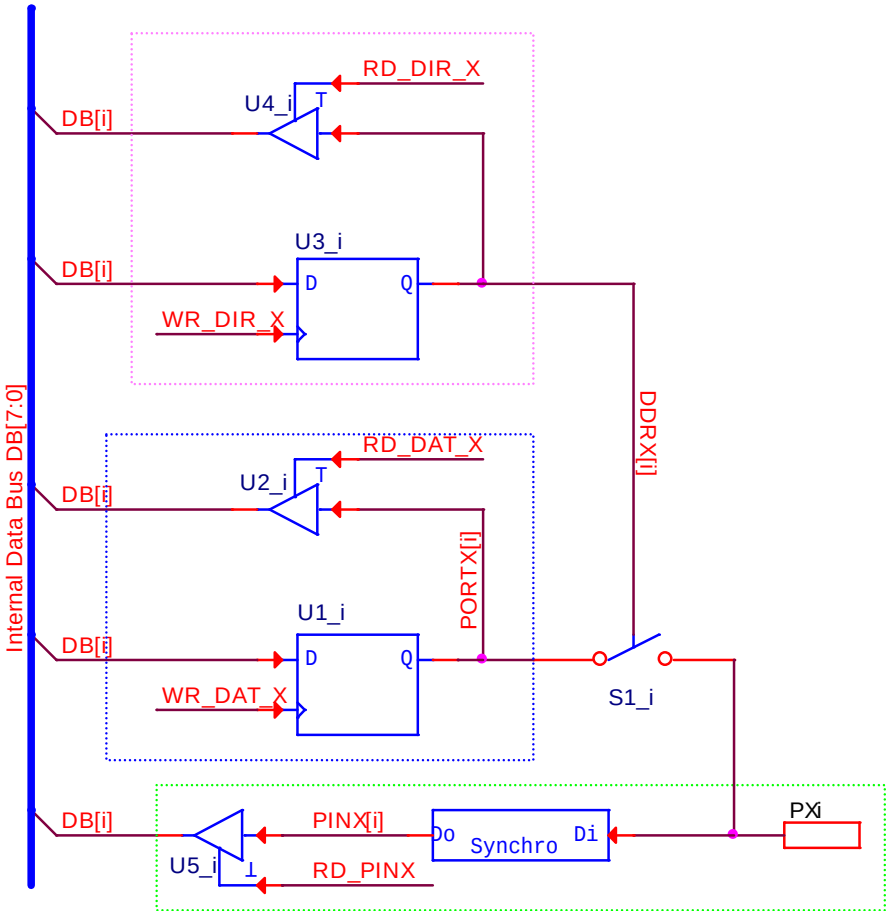


figura 4

In figura 4 în locul comutatorului cu două căi s-a folosit un comutator cu o singură cale deoarece se implementează mai simplu. Comutatorul cu o singură cale este un driver 3-state. Cu acest tip de comutator portul de intrare este tot timpul conectat la pin.

Toate cele 32 de structuri I/O sunt identice din puncte de vedere al funcţiei de bază şi anume digital I/O. Cele 32 de structuri sunt împărţite în 4 grupe de 8 biţi deoarece la procesorul AVR lăţimea căii de date este de 8 biţi. Nu există diferenţe hardware între grupe sau între membrii unui grupe. În figura 4 este prezentată structura asociată pinului PX*i*. X este una din literele A, B, C sau D iar *i* este o cifra în intervalul 0..7. Schema detaliată a acestei structuri este prezentată în capitolul 12 din documentaţia ATmega16. Pentru fiecare pin există însă şi o funcţie alternativă, funcţie ce depinde de pin. În acest moment nu vom trata funcţia alternativă.

Pinul PX*i* din figura 4 este controlat de două porturi procesor de tip buffer şi un port procesor de intrare: PORTX*i*, PINX*i* şi DDRX*i*.

- **PORTX*i*** este un bit dintr-un port procesor de tip buffer, aşa cum s-a discutat anterior. PORTX*i* este semnalul OUT din figura 3.
- **PINX*i*** este un bit dintr-un port procesor de intrare. PINX*i* este semnalul IN din figura 3. În structura portului de intrare remarcăm un bloc care nu apare anterior, şi anume blocul de sincronizare. Acest bloc are rolul de a reduce probabilitatea de apariție a fenomenului de **metastabilitate**. Metastabilitatea a fost discutată la curs. Ceea ce trebuie reținut este efectul secundar al blocului de sincronizare şi anume **o întârzierea două pulsuri de ceas** a semnalului aplicat pe pinul PX*i*. Mai precis, citirea pinului PX*i* nu generează valoarea curentă a semnalului aplicat pe acest pin, ci valoarea care era pe acest pin acum două pulsuri de ceas.
- **DDRX*i*** este un port procesor de tip buffer şi controlează comutatorul S1_i din figura 4. DDR este abrevierea lui **Data Direction Register**. Comutatorul S1_i este de fapt un driver 3-state dar a fost desenat ca un comutator pentru ca rolul sau să fie mai uşor de înțeles.

Semnalele WR_DAT_X, RD_DAT_X, WR_DIR_X, RD_DIR_X şi RD_PIN sunt generate de decodificatorul de adrese.

În prelegerea 2 s-a definit portul ca blocul prin care o entitate comunică cu exteriorul. În cazul unui microcontroler se pot evidenţia două entităţi: procesorul din interiorul microcontrolerul şi microcontrolerul însuşi. Din acest motiv apar două categorii de porturi: porturile procesor discutate în prelegerea 2 şi porturile microcontroler. Porturile procesor se vor mai numi registre IO iar denumirea de porturi se va folosi cu precădere pentru porturile microcontroler. Porturile A, B, C şi D ale microcontrolerului se mai numesc porturi digitale iar porturile PORTX, PINX şi DIRX sunt porturi procesor sau registre IO.

Fiind porturi buffer, PORTX şi DDRX pot fi atât scrise cât şi citite în orice moment. Deoarece PINX este port de intrare are sens doar citirea acestuia. În exemplele următoare vom folosi portul digital A format din PORTA, PINA şi DDRA. Adresele de port (adică adresa folosită în instrucțiunile IN şi OUT) pentru portul digital A sunt PINA=0x19, DDRA=0x1A şi PORTA=0x1B, Codul în limbaj de asamblare pentru operaţiile care se pot executa **întotdeauna** cu portul digital A sunt:

```
;scriere PORTA cu valoarea din R0. Bitul i din R0 se scrie în bistabilul U1_i din
;figura 4 .
out 0x1B, R0

;citirea lui PORTA în R1. Bitul i din U1 se citeste prin driverul 3-state U2_i.
in  R1, 0x1B

;scriere DDRA cu valoarea din R0. Bitul i din R0 se scrie în bistabilul U3_i.
out 0x1A, R0

;citirea lui DDRA în R1. Bitul i din U3 se citeste prin driverul 3-state U4_i.
in  R1, 0x1A

;citirea stării pinilor PA. Starea pinului PAi se citeste prin driverul U5_i.
in  R2, 0x19
```

Folosirea adreselor porturilor duce la un cod greu de urmărit. Codul devine mult mai clar dacă în loc de numere se folosesc nume. Pentru acesta este nevoie de directiva .EQU care are în asamblare rolul lui #define din C:

```
.EQU   PORTA = 0x1B
.EQU   DDRA  = 0x1A
.EQU   PINA  = 0x19
```

```
;scriere PORTA
out PORTA, R0

;citirea lui PORTA
in  R1, PORTA

;scriere DDRA
out DDRA, R0

;citirea lui DDRA.
in  R1, DDRA

;citirea stării pinilor PA.
in  R2, PINA
```

În C numele celor trei registre dintr-o grupă este identic cu numele folosit în datele de catalog. Pentru grupa A acestea sunt PORTA, PINA, DDRA. Aceste nume sunt declarate în headerul io.h. Codul în limbajul C pentru operaţiile care se pot executa **întotdeauna** cu portul digital A sunt:

```
#include <avr/io.h>
//adresele porturilor sunt definite în io.h
...
   unsigned char data;

   //scriere PORTA
   PORTA=data;

   //citirea lui PORTA
   data=PORTA

   //scriere DDRA
   DDRA=data;

   //citirea lui DDRA.
   data=DDRA;

   //citirea stării pinilor PA.
   data=PINA;
```

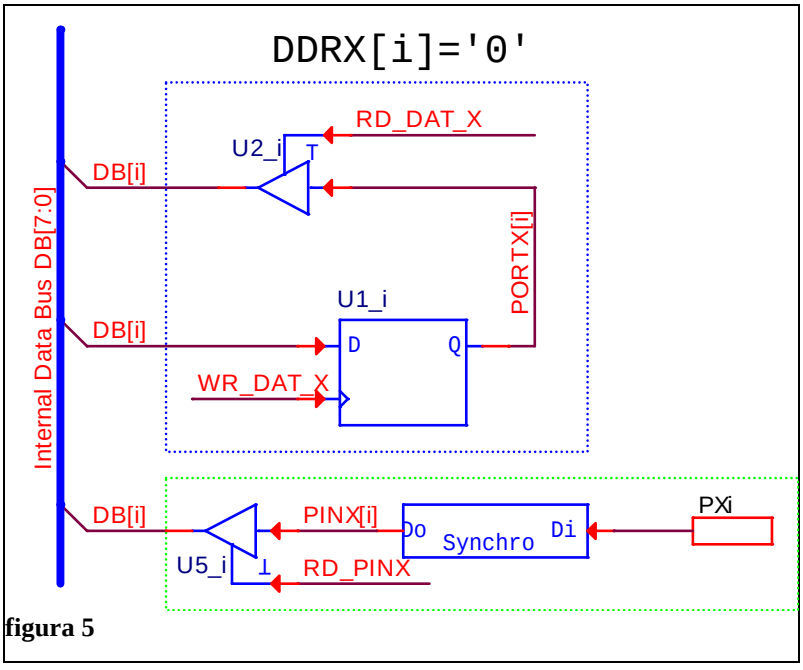


figura 5

Dacă DDRX*i* este ‚0' comutatorul S_i este deschis iar schema din figura 4 devine echivalentă cu schema din **Error! Reference source not found.**. Valoarea logică a pinului PX*i* este stabilită din exterior şi poate fi citită prin intermediul repetorului 3-state U5_i, ceea ce transformă acest repetor într-un **port microcontroler de intrare**. Pentru portul de intrare trebuie ţinută seamă de întârzierea două pulsuri de ceas introdusă de blocul de sincronizare. Suplimentar, PORTX*i* funcţionează ca o locaţie de memorie, fără ca valoarea înscrisă în bistabilul U1 să apară pe pinul PX*i*.

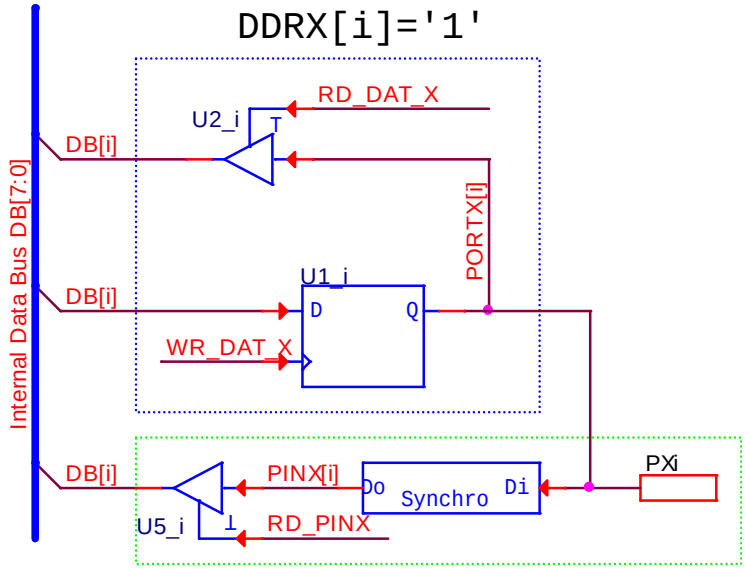
Deşi toate cele 5 operaţii discutate anterior sunt posibile, codul pentru configurarea şi utilizarea portului digital A ca port de intrare conține doar 2 operații. Acest cod este prezentat în continuare:

```
//Portul A este port de intrare
#include <avr/io.h>
...
unsigned char data;

   //în secţiunea de iniţializare
   DDRA=0b00000000; //sau DDRA=0;
...
   //citire port A
   data=PINA;

...
```

Dacă DDRX*i* este ‚1' comutatorul este închis iar schema din figura 4 devine **echivalentă** cu schema din figura 6. În figură ieşirea Q a bistabilului U1_i se conectează la pinul PX*i*, ceea ce transformă acest portul digital într-un **port de ieşire al microcontolererului**. Deşi toate cele 5 operaţii discutate anterior sunt posibile, codul pentru configurarea şi utilizarea portului digital A ca port de ieşire conține doar 3 operații. Codul se află în partea dreaptă a figurii.




```
//Portul A este port de iesire
#include <avr/io.h>
...
unsigned char data;

   //în secţiunea de iniţializare
   DDRA=0b11111111;//sau DDRA=0xff;
...
   //scriere port A
   PORTA=data;
...
   //citire port A
   data=PORTA;
...
```

figura 6

Din schemă se observă că data înscrisă în bistabilul U1_i poate fi citită şi prin intermediul repetorului 3-state U5_i cu *data = PINA*. Dacă vreodată este necesară citirea prin intermediul lui U5_i, trebuie ţinută seamă de întârzierea de două pulsuri de ceas introdusă de blocul de sincronizare. Practic, data scrisă în PORTX nu poate fi citită mediat prin PINX, ci trebuie aşteptat două instrucţiuni.

Chiar dacă direcţia se stabileşte pentru toată grupa X cu o singură instrucţiune, aceasta nu înseamnă că toți pinii sunt obligatoriu de IN sau toți sunt obligatoriu de OUT. De exemplu, dacă DDRA=0b1111010 atunci PA0 şi PA2 sunt pini de intrare iar PD1, PD3, PD4, PD5, PD6 şi PD7 sunt pini de ieşire.

Pentru porturile procesor nu este posibil accesul în scriere sau citire la nivel de bit, dar există instrucţiuni care pot set sau reseta individual fiecare bit.