7.Interfețe integrate în microcontrollere

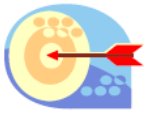
Cuprins

Cuprins modul

7.1. Microcontrollere
7.2. Unitatea centrală şi memoria
7.3. Timerul
7.4. Interfață de comunicații seriale asincrone
7.5. Interfață de comunicații seriale sincrone
7.6. Interfața CAN
7.7. Ceasul de gardă
7.8. Generator PWM
7.9. Convertor analog digital
7.10. Proiectarea sistemelor cu MC în vederea siguranței în exploatare
7.11. Medii de programare şi exemple

Introducere

Modulul "**Interfețe integrate în microcontrollere**" face o prezentare generală a arhitecturii microcontrollerelor, începând cu unitatea centrală, memoria şi continuând cu cele mai utilizate interfețe. Sunt abordate interfața de comunicații asincronă, sincronă şi o interfață avansată cu comunicații prin cadre de date, interfața CAN. Sunt apoi prezentate sumar şi alte interfețe şi module interne importante, cum sunt timer-ul, ceasul de gardă, generatorul PWM şi convertorul analog digital. La sfârşitul materialului sunt prezentate câteva aplicații cu microcontrollere făcute de studenți.



Obiective

După parcurgerea acestui modul studenții îşi vor completa informațiile despre microcontrollere. Analiza caracteristicilor şi a funcționării interfețelor interne îi va ajuta să înțeleagă că particularitățile fiecărei interfețe o recomandă pentru un anumit tip de aplicații, de o anumită complexitate şi cu anumite cerințe de performanță.
Obiective specifice:
1.Cunoaşterea unor tipuri de transmisii seriale şi interfețe seriale şi paralele ca structură, protocol şi interfețe tipice
2.Învățarea unor interfețe tipice interne din microcontrollere
3.Înțelegerea noțiunilor prin exemplificări practice

Durata medie de studiu individual

Durata medie de studiu individual este de 2 ore.

7.1. Microcontrollere

Microcontrollerul (MC) are schema bloc generală simplificată din figura 7.1. În această schemă sunt puse în evidență interfețele, numite şi dispozitive de intrare ieşire I/O.

Unitatea centrală execută instrucțiunile din memoria program, pe care le primeşte prin magistrala de date. Structura Harvard este posibilă şi răspândită la MC pentru că de regulă instrucțiunile sunt stocate în memoria ROM iar datele în cea RAM. Fiecare MC are un controller de întreruperi şi unele au controller de DMA care admit atât intrări din exterior cât şi de la modulele interne. Modulele de I/O pot fi seriale sau paralele. Fiecare modul transferă date cu exteriorul prin intermediul registrului de date (RD). Modulul este comandat (configurat) de unitatea centrală prin intermediul unui registru de comenzi (RC) şi se poate citi starea modulului prin registrul de stare (RS), prin care se pot şi cere întreruperi. Registrele modulelor de I/O pot fi văzute de UC ca locații de memorie (la familia Motorola) sau ca dispozitive de I/O într-un spațiu de adresare separat (MCS 51). De regulă structura de bază a familiei conține anumite interfețe considerate foarte importante (timer, canal serial UART) şi linii de I/O grupate în porturi paralele de uz general. Pe structura de bază se adaugă diferite tipuri de interfețe care împart liniile de I/O cu porturile paralele de uz general.

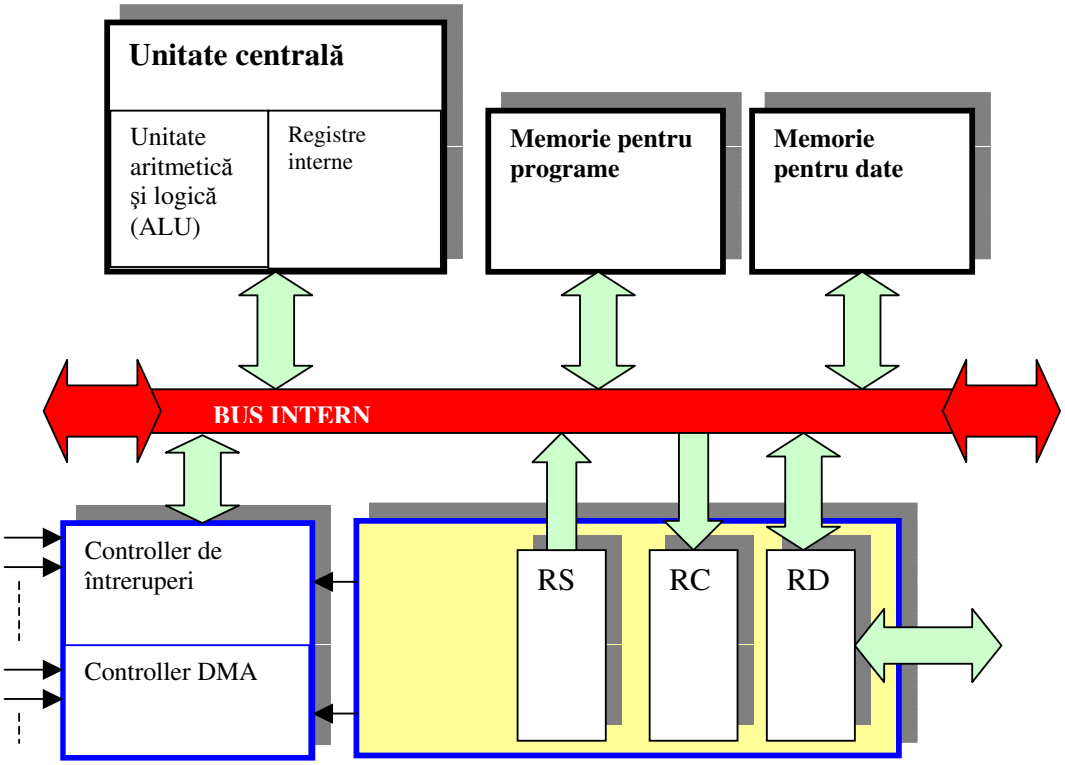


Figura 7.1. Schema bloc generală simplificată a unui microcontroller

Blocurile interne ale MC sunt legate între ele printr-o magistrală (bus) de date şi una de adrese. Mărimea acestor magistrale constituie una dintre caracteristicile cele mai importante ale unui MC.

Arhitectura Von Neuman prevede existența unui bus unic folosit pentru circulația datelor şi a instrucțiunilor. Când un controller cu o astfel de arhitectură adresează memoria pe linia de date se pune întâi codul instrucțiunii apoi conținutul memoriei, accesul fiind realizat în 2 paşi, deci destul de lent. Arhitectura Harvard prevede un bus separat pentru date şi instrucțiuni. Când instrucțiunea este pusă pe busul de instrucțiuni, datele de la instrucțiunea anterioară sunt pe busul de date. Structura MC este mai complexă, dar performanțele de viteză sunt mai bune. Magistrala de date şi cea de adrese pot fi separate sau multiplexate. Magistralele pot să nu fie scoase în exterior (Motorola 6805) sau pot fi scoase în exterior direct (MCS 51) sau multiplexate (MC pe 16 sau 32 de biți).

7.2. Unitatea centrală şi memoria

Unitatea centrală este formată din ALU şi un set de regiştri interni, figura 7.2., similari unor locații de memorie, folosiți pentru memorarea unor date des folosite sau pentru programarea unor anumite funcții. Diferitele familii de MC folosesc seturi diferite de regiştri. Există însă câțiva regiştri comuni:

1. **A** (Accumulator) registrul acumulator care este folosit deseori pentru a stoca un operand şi rezultatul unei operații aritmetice.
2. **I** registru de index, folosit la adresări indirecte.
3. **S** registru de stare, care conține indicatorii de stare: Carry, Zero etc.
4. **PC** (Program Counter) este stocată adresa următoarei instrucțiuni de executat. După un RESET (inițializarea MC), registrul PC se încarcă dintr-o locație de memorie numită vector de reset. Această locație conține adresa primei instrucțiuni de executat. După execuția acestei prime instrucțiuni, PC se incrementează.
5. **SP** (Stack Pointer) conține indicatorul de stivă. Stiva este o memorie de tip LIFO, în care ultimul octet stocat este primul scos din memorie. Conținutul acestui registru stabileşte adresa din memorie unde este definită stiva.

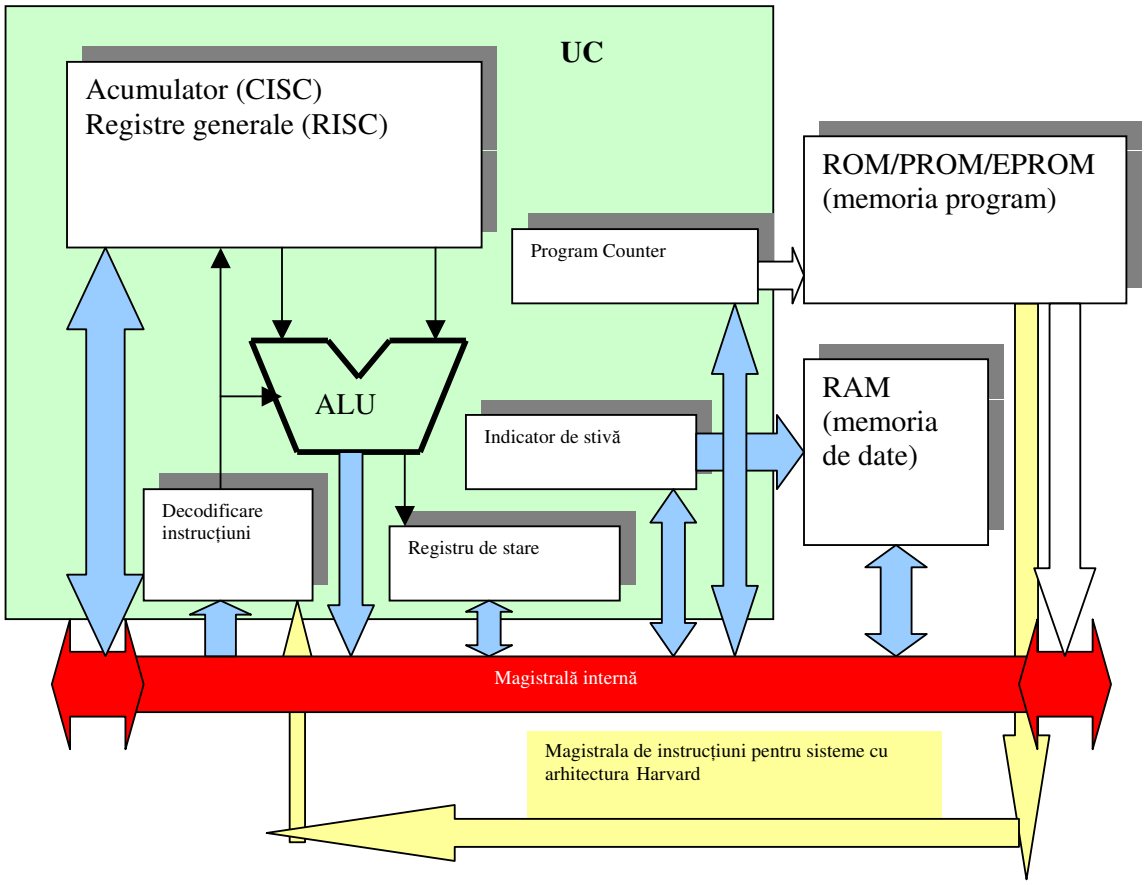


Figura 7.2. Schema bloc a unei unități centrale tipice

Tipurile de memorii utilizate în prezent la microcontrollere sunt:

1. Memoria ROM (Read Only Memory) este cea mai ieftină şi simplă memorie şi se foloseşte la stocarea programelor în faza de fabricație. Unitatea centrală poate citi informațiile, dar nu le poate modifica.
2. Memoria PROM (Programmable Read Only Memory) este similară cu memoria ROM, dar ea se poate programa de către utilizator. După posibilitățile de ştergere, această memorie poate fi de mai multe feluri:

- OTP (One Time Programmable, PROM) este de fapt o memorie EPROM, dar *chipul* a fost capsulat într-o capsulă de material plastic fără fereastră, care este mult mai ieftină. Memoria nu se poate şterge sau reprograma. Prețul unui MC cu OTP este mic, viteza este bună, dar aplicațiile sunt lipsite de flexibilitate.
- Memorii care pot fi şterse electric de către unitatea centrală, în timpul funcționării. Ştergerea este selectivă, iar pentru reînscriere trebuie făcuți mai mulți paşi. Astfel de memorii sunt cele EEPROM (Electrically Erasable Programmable Read Only Memory) şi memoria FLASH EPROM.

3.Memoria RAM (Random Access Memory) este o memorie volatilă care poate fi citită sau scrisă de unitatea centrală. Locațiile din RAM pot fi accesibile în orice ordine. Pe chip, memoria RAM ocupă mult loc şi implicit costurile de implementare sunt mari. De aceea un MC include de obicei puțin RAM.

Observații:

1. Stocarea programelor în memorii nevolatile permite ca MC să fie programat fără a fi scos din circuitul în care funcționează (Field Programming/ Reprogramming).
2. Producătorii recomandă ca la producții de volum mare să se folosească memoria ROM, care se înscrie la fabricant, cu mască, la producții de volum mic să se folosească memoria OTP (PROM) iar pentru prototipuri să se folosească memoria EEPROM sau FLASH.
3. Unele familii de MC tratează spațiul de intrare ieşire ca şi memoria, iar altele au spații diferite de adresare pentru memorie şi spațiul I/O. Tratarea unitară a acestor spații are avantajul simplității dar limitează numărul de locații de memorie adresate. Tratarea unitară este justificată de asemănările existente între stocarea unui bit în memorie sau într-un latch de I/O.

7.3. Timerul

Timerul este modulul intern care realizează funcțiile de timp, fiind unul dintre modulele cele mai importante şi des folosite. Modulele timer sunt foarte diferite ca şi funcționalități şi complexitate, aspect de care trebuie ținut cont la alegerea microcontroller-ului pentru aplicațiile sensibile la timp. Schema bloc a modulului timer este dată în figura 7.3.

Tactul pentru timer poate fi ales tactul sistem sau un tact de la un pin extern. Tactul poate fi divizat de un număr programabil de ori cu un numărător de prescalare (timer de prescalare). Comanda funcționării timerului se face cu un registru de control şi stare.

Timerul este format din 3 module, a căror caracteristică este numărul de biți:

- numărător (timer)
- registru de încărcare
- registru de captură

Modurile de lucru sunt următoarele:

1. mod numărător, în care Timerul numără tactul de la intrare şi la depăşire înscrie un bit în registrul de stare sau se poate cere o întrerupere. Dacă tactul este extern înseamnă că se face o numărare a evenimentelor externe. De regulă frontul activ se poate programa, adică să se facă o incrementare a numărătorului la front pozitiv sau negativ.

2. mod captură, în care un pin exterior poate comanda oprirea timerului şi încărcarea valorii la care a ajuns în registrul de încărcare. Acest mod poate fi folosit pentru măsurarea unei perioade de timp.

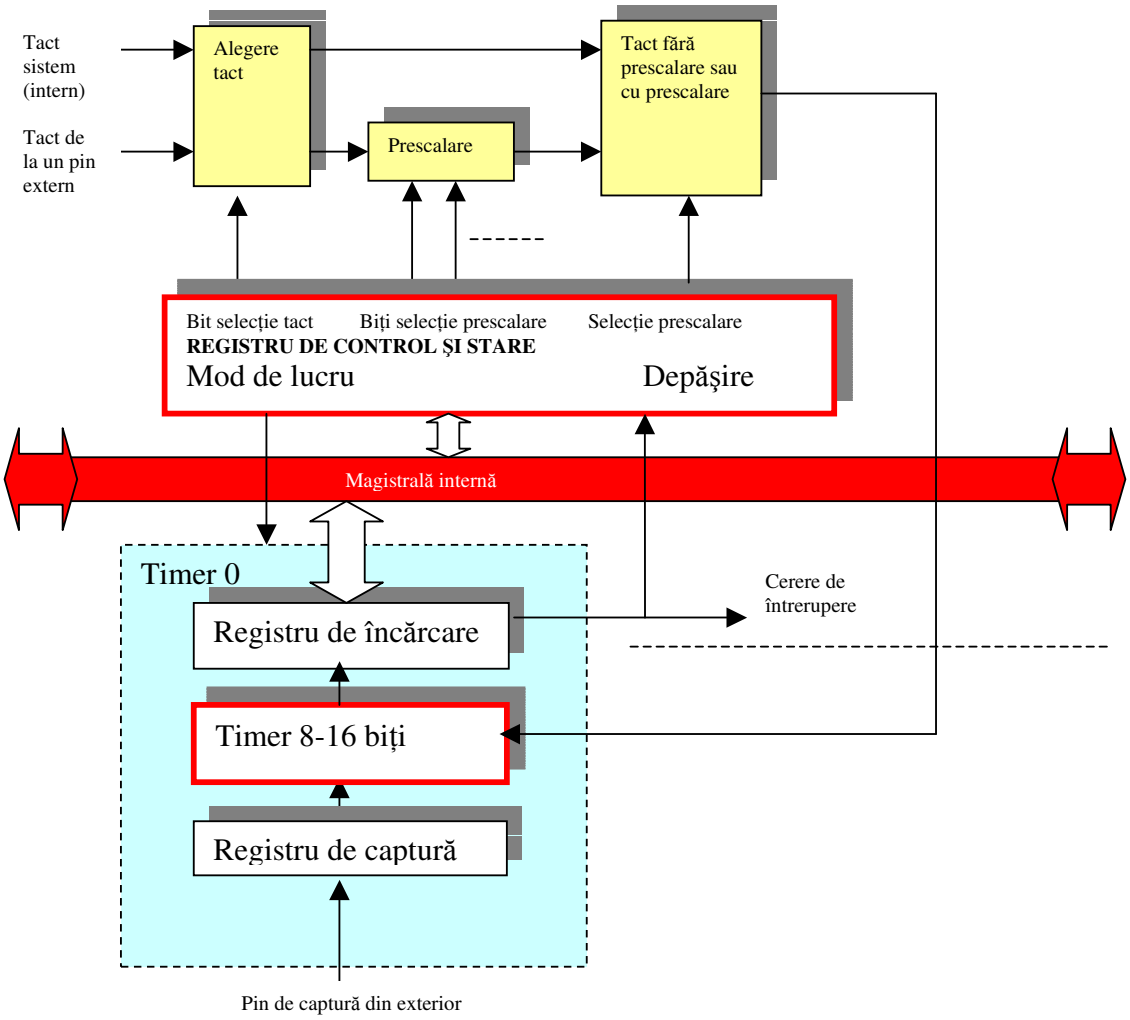


Figura 7.3. Schema boc a modulului timer

7.4. Interfață de comunicații seriale asincrone

Portul de comunicații seriale asincrone este numit UART (Universal Asyncronous Receiver Transmitter), iar Motorola îl numeşte pentru microcontrollerele proprii port SCI (Serial Communications Interface). Schema bloc a interfeței este dată în figura 7.4.

Caracterele seriale sunt transmise sau recepționate serial în registrele de transmisie sau recepție. La recepția unui caracter, acesta se încarcă în bufferul de recepție şi se cere o întrerupere. La emisie, un caracter se introduce în bufferul de transmisie de unde este

trecut în registrul de deplasare şi se transmite serial, cerându-se şi o întrerupere. Ceasul poate fi selectat intern sau extern. Dacă este selectat intern, el se formează din tactul sistemului cu o divizare printr-un numărător de 16 biți (prescalare) şi apoi un numărător de 11 biți. Comanda USART se realizează cu un registru de stare şi control

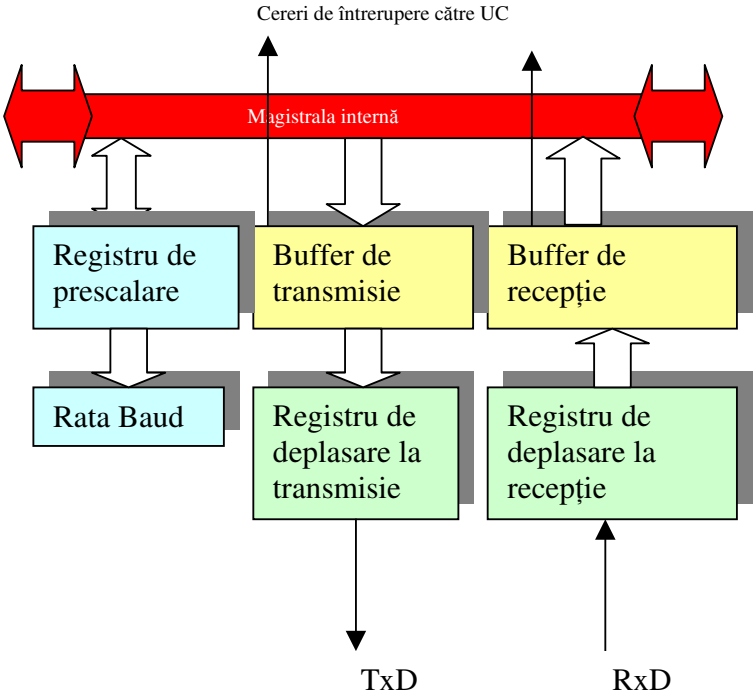


Figura 7.4. Schema bloc a interfeței de comunicații seriale asincrone

7.5. Interfață de comunicații seriale sincrone

Cu portul SPI (**Serial Peripheral Interface**) se poate realiza o comunicație sincronă simplă, folosită de regulă pentru a transfera date între circuite pe aceeaşi placă cu MC. Un transfer bidirecțional necesită 3 pini, unul dintre ei fiind alocat ceasului de transmisie generat de masterul SPI. Cu SPI se pot realiza transferuri şi între MC. Transferurile pot fi full duplex. Schema bloc a conectării unor dispozitive prin interfața serială sincronă este dată în figura 7.5.

Numai un master SPI poate iniția un transfer. Masterul scrie un octet în registrul de transmisie SPI de undele datele merg într-un registru de deplasare care le serializează şi le transmite cu ceasul de transmisie. Transmisia se termină după 8 tacte. În slave datele intră în registrul de deplasare cu tactul de recepție, acelaşi cu cel de transmisie. Când au intrat 8 biți, caracterul intră în registrul de date. Pentru a se evita erorile de viteză (sau de

suprascriere)(Overrun) trebuie ca octetul din registrul de date să fie citit înainte ca un alt octet să fie transmis din registrul de deplasare.

Pinii au următoarea semnificație:

- SCK (Serial Clock) este ieşire de tact pentru sincronizare;
- MOSI (Master Output Slave Input) este ieşirea serială pentru MASTER;
- MISO (Master Input Slave Output) este intrarea serială pentru MASTER;
- /SS (Slave Select) selectează circuitul SLAVE şi protejează MC dacă două circuite sunt master. Acest semnal activ dezactivează la celălalt port SPI modul master.

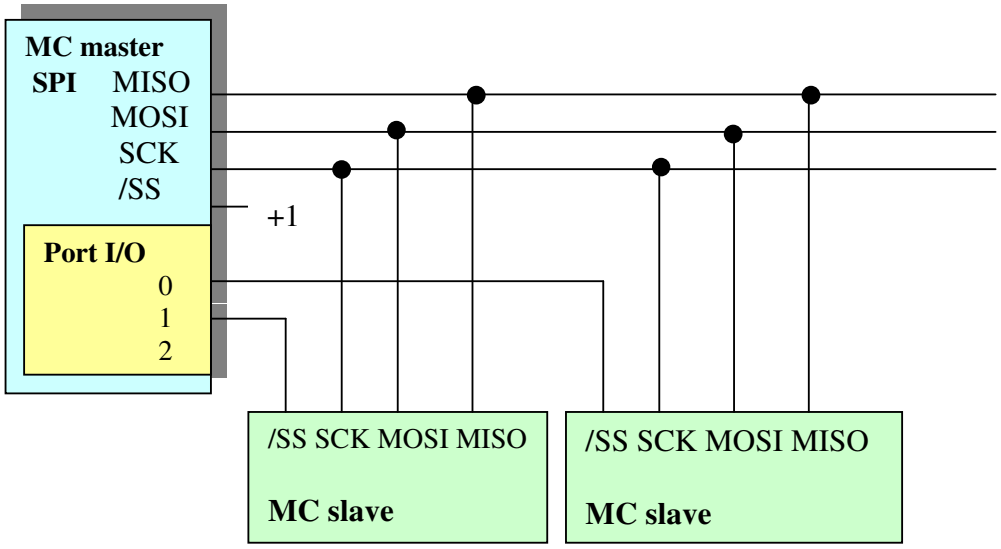


Figura 7.5. Schema bloc a conectării unor dispozitive prin interfeța serială sincronă

Dacă dispozitivul master lucrează cu mai multe dispozitive slave atunci semnalul /SS nu este suficient şi selecția dispozitivului slave se face cu linii de ieşire dintr-un port auxiliar, ca în figură.

7.6. Interfața CAN

Protocolul CAN (**Controller Area Network)** a fost definit de BOSCH în 1991 pentru utilizarea pe o magistrală la autoturisme, unde să îndeplinească condiții specifice: procesare în timp real, fiabilitate într-un mediu perturbat şi preț mic.

La transmisia CAN datele sunt codificate pentru a fi trimise pe linie în cod NRZ, iar la fiecare grup consecutiv de 5 biți cu aceeaşi valoare logică se introduce un bit (deci o tranziție) care se extrage la decodificare. Nivelele pe linie sunt numite dominant (0) şi recesiv (1).

Transferul de date prin CAN se face cu cadre (blocuri de date) care sunt citite de toate dispozitivele cuplate la CAN dar sunt reținute de acestea doar acelea care conțin adresa dispozitivului.

Fiecare dispozitiv CAN recepționează toate cadrele şi dispune de un filtru de acceptanță care selectează cadrul cu adresa proprie a dispozitivului. în configurație se pot adăuga noi dispozitive, cu adresă proprie, fără nici un efort. Dacă cadrul recepționat este eronat, toate dispozitivele CAN trimit un cadru de control care indică o eroare. Fiecare cadru eronat incrementează în dispozitiv un numărător de erori (care este decrementat de cadrele valide). Un număr de erori mai mare de o anumită limită produce decuplarea dispozitivului de la magistrala CAN. Schema bloc a interfeței CAN este dată în figura 7.6.

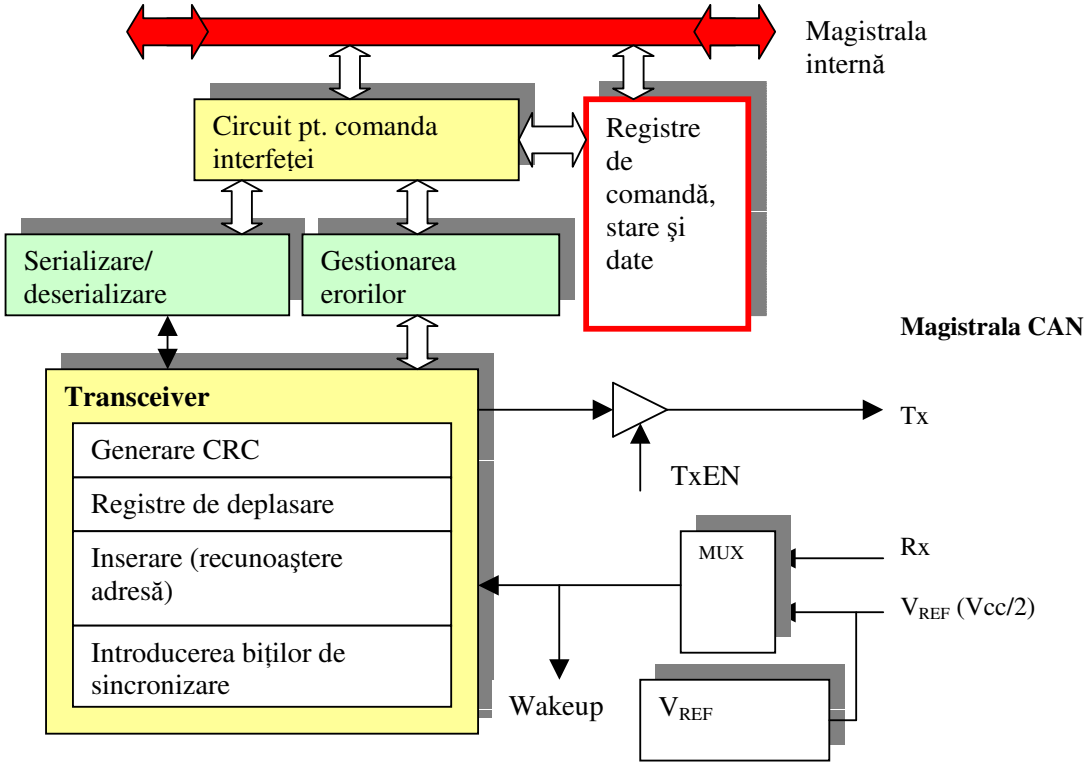


Figura 7.6. Schema bloc a interfeței CAN

7.7. Ceasul de gardă

Ceasul de gardă este un timer care poate fi programat să numere un tact care provine de la un registru de prescalare, figura 7.7. Dacă numărătorul ajunge la capăt, semnalul de depăşire declanşează un RESET al circuitului. Este sarcina programatorului să scrie în

registrul de control un cuvânt care va reinițializa numărătorul. În cazul în care MC nu mai este sub controlul programului, el va fi resetat de către ceasul de gardă

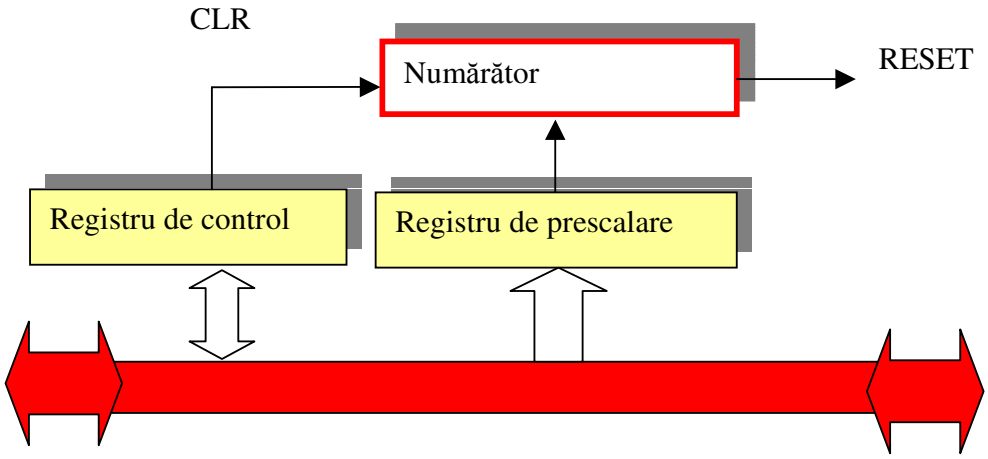


Figura 7.7. Ceasul de gardă

7.8. Generator PWM

Modulația impulsurilor în lățime (Pulse Width Modulation) are multe aplicații, mai ales în comanda motoarelor de curent continuu sau a surselor de alimentare. Din acest motiv, unele MC includ în structura lor un modulator PWM ca interfață distinctă. Un semnal PWM arată ca în figura 7.8.

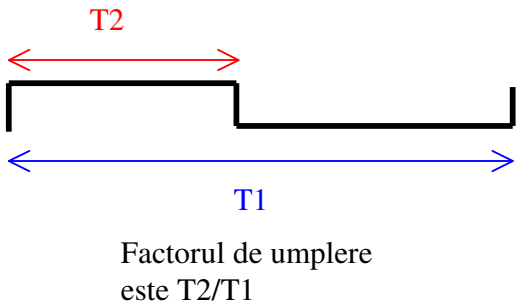


Figura 7.8. Semnal cu punerea în evidență a factorului de umplere

Schema bloc a unui generator PWM este reprezentat în figura 7.9. Frecvența de repetiție este programată cu un registru de prescalare care generează un ceas pentru un numărător de 8 biți. Conținutul numărătorului este comparat cu cel al registrului PWM, dacă este

mai mare ieşirea PWM este LOW, dacă este mai mic sau egal PWM este HIGH. Factorul de umplere poate fi astfel modificat între 0 şi 254/255.

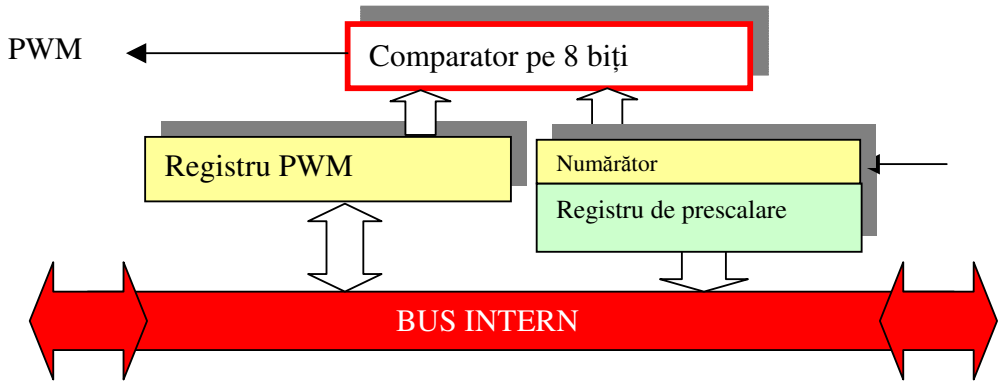


Figura 7.9. Schema bloc a unui generator PWM

7.9. Convertor analog digital

Schema bloc a unei interfețe tipice de conversie a semnalelor analogice în semnale digitale este dată în figura 7.10. Circuitul analogic de intrare constă într-un multiplexor analogic şi un convertor A/D de 8-10-12 biți cu aproximații succesive (sunt câteva MC cu convertor cu integrare, COP8). Tensiunea de referință pentru convertor şi masa analogică sunt conectate prin pini speciali. Convertorul este controlat de registrul de control, care selectează şi canalul de conversie. Terminarea conversiei este semnalizată cu un bit ACK tot în registrul de control, iar rezultatul conversiei este stocat în registrul de date.

O conversie poate fi declanşată în 3 feluri:
-start în operare normală şi reintrare în operare normală;
-start în operare normală apoi intrare în mod inactiv (Idle);
-intrare în mod inactiv şi declanşarea unei conversii din exterior printr-un pin exterior.

De regulă modul de conversie poate fi:
-conversie singulară
-conversie continuă

Cu registrul de control se poate programa:
-selecția canalului analogic dorit de la intrare la convertorul AD;
-se poate programa ca o conversie să fie declanşată de pinul extern ;
-se poate declanşa o conversie;
-conține un bit care semnalează că s-a terminat conversia, care poate solicita o cerere de întrerupere;

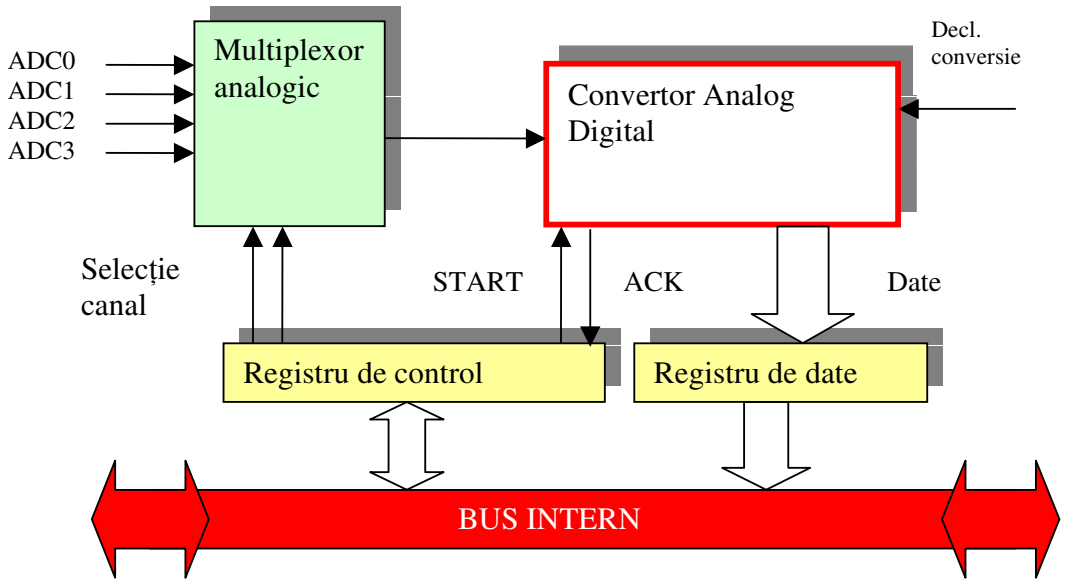


Figura 7.10. Schema bloc a unui convertor analog digital

7.10. Proiectarea sistemelor cu MC în vederea siguranței în exploatare

Pot apare 2 categorii de probleme: aplicația poate genera perturbații (conduse sau radiate) sau poate fi susceptibilă la perturbații (conduse sau radiate). Descoperirea unor **probleme de EMI** în timpul producției aplicației poate fi costisitor deoarece s-ar putea să fie necesară reproiectarea aplicației, de aceea este necesar să se **proiecteze în vederea EMC**.

Perturbațiile sunt generate de armonicile semnalelor digitale din circuit. Ele pot fi radiate de buclele de cablaj care se comportă ca şi antene sau sunt conduse spre sursa de alimentare. Orice cale inductivă sau capacitivă pe traseul acestor armonici poate provoca vârfuri de tensiune sau căderi de tensiune. Pentru un sistem cu un MC, perturbațiile sunt generate de regulă de cablaj, deoarece circuitele integrate au dimensiuni prea mici pentru a putea emite. Semnalul cu frecvența cea mai mare este tactul sistemului, generat cu un circuit oscilant cu cuarț. Datorită faptului că forma semnalului este apropiată de forma sinusoidală, conținutul de armonici este mic. Dacă tactul este adus din exterior, este nevoie de mare grijă pentru a reduce buclele de circuit emisive.

Pentru un sistem care are memorii externe cuplate la MC, liniile de transfer pot fi emisive, deoarece frecvențele de tranziție sunt mari.

Prin metodele de **programare defensivă** se poate îmbunătăți mult siguranța în funcționare, fără nici un hardware suplimentar. Câteva din cele mai eficiente metode sunt:

-reîncărcarea periodică a registrelor care comandă pinii de I/O şi a celor mai importante registre. Pinii de I/O sunt legătura MC cu exteriorul, de aceea ei sunt supuşi perturbațiilor. Readucerea lor la nivele corecte micşorează probabilitatea ca o perturbație să se propage în circuit.

-citirea repetată a semnalelor de intrare micşorează riscul unei citiri greşite. De exemplu citirea unui pin care este legat la o tastă de 3 ori la rând la intervalul de timp normal pentru 3 citiri succesive, dacă s-a citit aceeaşi valoare, elimină posibilitatea unei perturbații.

-dacă există locații în RAM nefolosite, după fiecare etapă de rulare a programului se scrie un bit în RAM. Înainte de rularea unei rutine critice se verifică valoarea stocată în RAM şi rutina se execută doar în cazul în care valoarea din RAM este corectă.

-dacă într-o aplicație există memorie nefolosită, aceasta se umple cu instrucțiuni de salt într-un loc cunoscut pentru ca un salt neprevăzut în memorie datorat unei perturbații să fie anulat de saltul în locul cunoscut, cu o anumită probabilitate.

7.11. Medii de programare şi exemple

Pentru a realiza programe, pentru fiecare familie de microcontrollere există medii software care uşurează munca de programare, testare şi înscriere a codului în MC. Astfel câteva exemple de capturi de ecrane ale unor medii de dezvoltare sunt date în figura 7.11.

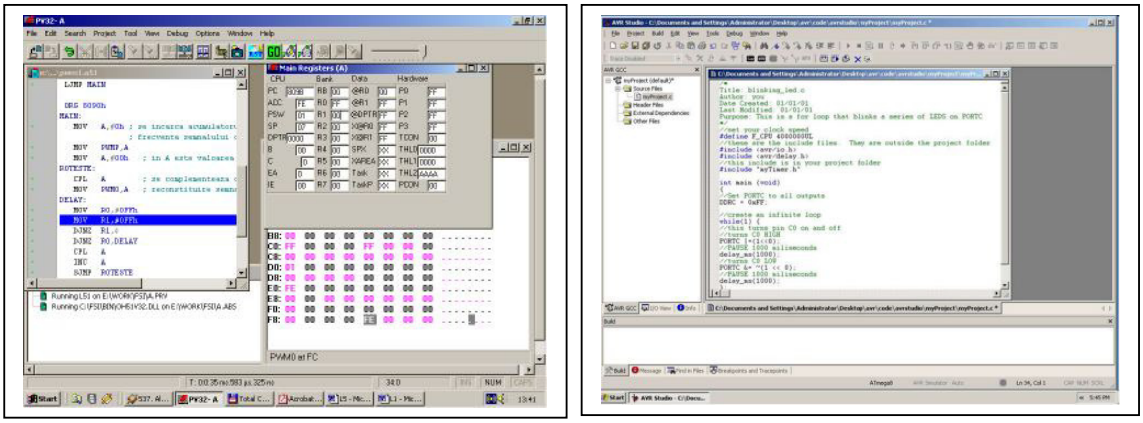


Figura 7.11. Mediu de dezvoltare pentru MCS 51 Franklin software (stânga) şi mediu de dezvoltare pentru ATMEL RISC, AVR Studio, (dreapta)

Mediul de dezvoltare MPLAB pentru PIC este arătat în figura 7.12. stânga şi o aplicație cu microcontroller realizată de un student la proiect este dată în dreapta.

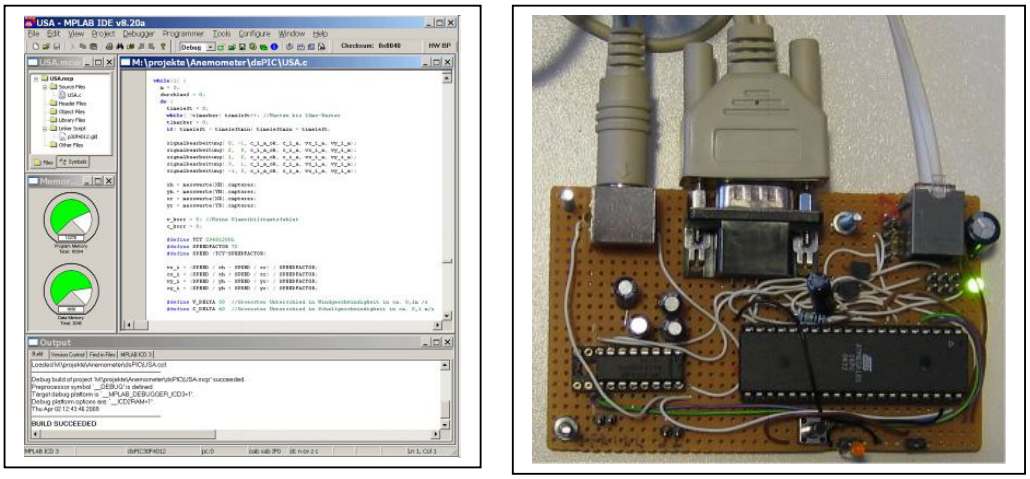


Figura 7.12. Mediul de dezvoltare MPLAB pentru PIC (stânga) şi o aplicație cu microcontroller (dreapta)

Se poate observa în aplicația cu microcontroller faptul că realizarea a fost la nivel de amator, fără construirea unui cablaj, totuşi aplicația a funcționat, a fost simplu de realizat şi studentul a avut ocazia să învețe multe lucruri. Pe placă, al doilea circuit este un MAX232 pentru adaptarea de nivel de tensiune de la interfața serială asincronă UART la RS232.

O aplicație realizată pentru proiectul de diplomă este un sistem mobil echipat cu un senzor de gaz, un modul Arduino cu microcontroller, drivere pentru motoare de curent continuu şi modul Bluetooth, figura 7.13 stânga. Comanda de la distanță se face cu un telefon mobil, interfața grafică a programului fiind arătată în dreapta.

Figura 7.13. Aplicație cu microcontroller

Rezumat

Această unitate de învățare prezintă cele mai utilizate interfețe integrate în microcontrollere. Pentru a putea înțelege funcționarea interfețelor materialul începe cu o prezentare a unității centrale şi a tipurilor de memorii care intră în structura microcontrollerelor. Sunt prezentate apoi interfețele de comunicații, începând cu cea mai simplă-interfața serială asincronă, apoi cea sincronă şi mergând până la interfața CAN. Sunt prezentate în continuare şi alte interfețe importante cum sunt timer-ul, convertorul analog digital, ceasul de gardă şi generatorul PWM. Materialul se termină cu câteva indicații de proiectare a software-ului care să asigure stabilitatea operațiilor de intrare ieşire, o scurtă prezentare a câtorva medii de dezvoltare software şi câteva fotografii ale unor aplicații realizate de studenți.

Bibliografie

1. M. Romanca, P. Ogrutan, *Sisteme cu calculator incorporat. Aplicatii cu microcontrollere*, Editura Universitatii Transilvania Brasov, 2011, pag. 1-4 online la: http://vega.unitbv.ro/~ogrutan/Microcontrollere2011/3-usb-ieee1394.pdf
2. C. Gerigan, P. Ogruțan**,** *Tehnici de interfațare*, Ed. Transilvania Braşov, 2000, 315p., ISBN 973-9474-94-2, pag. 154-174, online la: http://vega.unitbv.ro/~ogrutan/ti/index.html
3. P.Ogrutan, *Microcontrollere si controllere grafice Fujitsu,* Ed. Universitatii Transilvania Brasov, 2006, 182 pag, ISBN 973-635-621-3, pag. 125-134, online: http://vega.unitbv.ro/~ogrutan/Microcontrollere%20Fujitsu/interfete3-112-136.pdf ,

Test de autoevaluare

1.Pe magistrala Harvard instrucțiunile şi datele circulă pe aceleaşi linii

R

☐ adevărat

☐ fals

I. Vezi pagina 3

2.Alegerea ca tact pentru timer tactul sistemului permite generarea unor perioade de timp precise, doar dacă tactul este generat cu un cuarț

R

☐ adevărat

☐ fals

I. Vezi pagina 5

3.Semnalul /SS la interfața SPI permite selectarea a cel puțin 4 dispozitive slave

R

☐ adevărat

☐ fals

I. Vezi pagina 8

4.La un generator PWM pe 10 biți factorul de umplere poate fi modificat între 0 şi 1023/1024

R

☐ adevărat

☐ fals

I. Vezi pagina 11

5.Ceasul de gardă are o structură şi funcționare asemănătoare cu:

R

- ☐ (a) interfața SPI
- ☐ (b) timer
- ☐ (c) interfața UART
- ☐ (d) interfața CAN

I. Vezi pagina 10

6.La un convertor A/D pentru ca unitatea central să nu perturbe conversia, o conversie se poate declanşa:

R

- ☐ (a) start în operare normală şi reintrare în operare normală;
- ☐ (b) start în operare normală apoi intrare în mod inactiv (Idle);
- ☐ (c) intrare în mod inactiv şi declanşarea unei conversii din exterior printr-un pin exterior
- ☐ (d) cu un bit din registrul de control

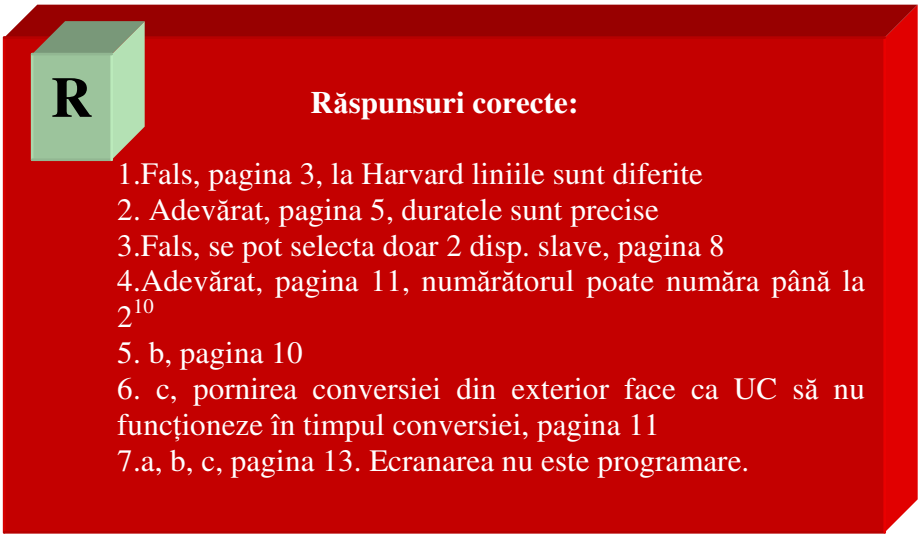
I. Vezi pagina 11

7.Exemple de programare defensivă sunt:

R

- ☐ (a) reîncărcarea periodică a registrelor
- ☐ (b) citirea repetată a semnalelor de intrare
- ☐ (c) memoria nefolosită se umple cu instrucțiuni de salt într-un loc cunoscut
- ☐ (d) ecranarea generatorului de tact

I. Vezi pagina 13


R
Răspunsuri corecte:
1.Fals, pagina 3, la Harvard liniile sunt diferite
2. Adevărat, pagina 5, duratele sunt precise
3.Fals, se pot selecta doar 2 disp. slave, pagina 8
4.Adevărat, pagina 11, numărătorul poate număra până la 2^{10}
5. b, pagina 10
6. c, pornirea conversiei din exterior face ca UC să nu funcţioneze în timpul conversiei, pagina 11
7.a, b, c, pagina 13. Ecranarea nu este programare.