12. Convertorul analog digital

Cuprins

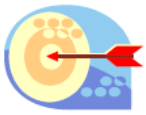
Cuprins Laborator 12

Cunoştinţe preliminare

Limbaj de asamblare MCS 51

Introducere

Această lucrare este destinată studenților care au înțeles programarea portului paralel, serial şi a timerului. Aceştia au ocazia să-şi aprofundeze cunoştințele prin programarea convertorului analog digital. Studenții care nu au reuşit să înțeleagă în totalitate modul de programare din lucrările precedente au ocazia să repete părți din lucrările anterioare. În această lucrare este prezentat convertorul analog digital şi registrul de programare. Un exemplu de program ajută studenții să înțeleagă programarea convertorului.



Obiective

Studenții vor înțelege funcționarea şi modul de programare a convertorului analog digital. Vor avea de asemenea ocazia să repete ce au învățat la portul serial, realizând un program care trimite valoarea digitală la calculatorul PC.

Durata medie de studiu individual

Durata medie de studiu individual este de 2 ore.

12.1. Funcționarea convertorului analog digital

Microcontrolerele Philips 8xC552 conțin un convertor A/D de 10 biți (figura 12.1), cu 8 canale analogice multiplexate, care lucrează prin metoda aproximațiilor succesive. Biții AADR2÷AADRO din ADCON selectează una din cele 8 intrări analogice (ADC7-ADCO). Pentru tensiunile de referință (Avref+, Avref-) şi de alimentare a circuitelor analogice (AVDD, AVss) sunt prevăzuți pini separați.

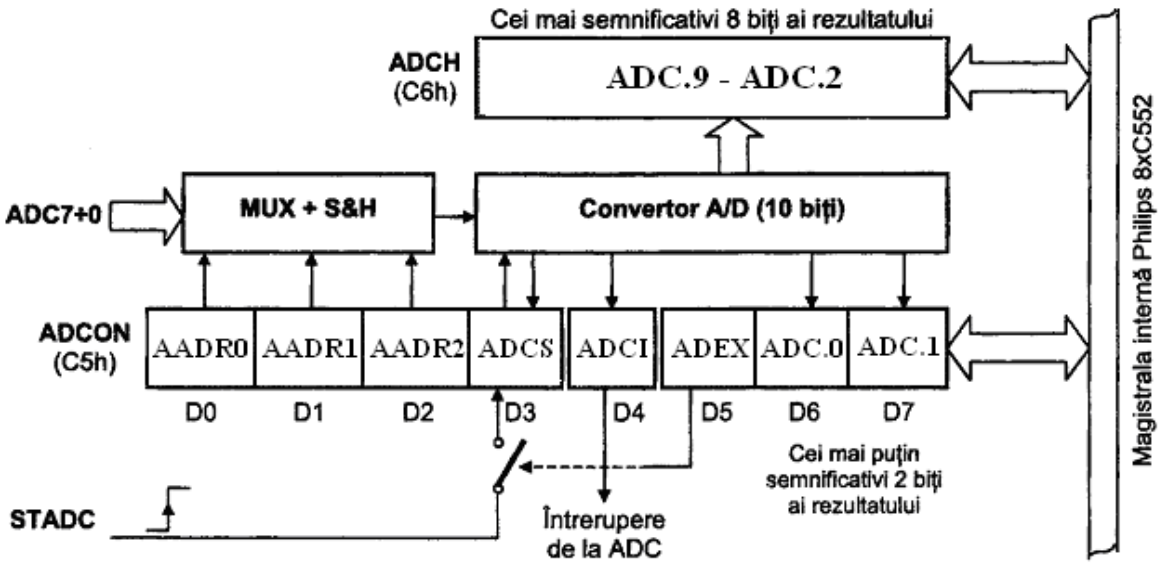


Figura 12.1 Sistemul de intrari analogice

O conversie poate fi declanşată:

1. numai prin software, prin setarea bitului ADCS (A/D Conversion Start/Status) din SFR ADCON (ADCON.3), dacă bitul ADEX (ADCON.5) = 0;
2. atât prin software, cât şi prin hardware, când ADCS este automat setat la detectarea unui front crescător pe linia STADC (STart A/D Conversion), dacă ADEX = 1. Pentru o corectă declanşare a unei conversii prin hardware, linia STADC trebuie să rămână cel puțin un ciclu maşină pe nivel "0", urmat apoi de cel puțin un ciclu maşină pe nivel "1".

ADCON

ADC.1	ADC.O	ADEX	ADCI	ADCS	AADR2	AADR1	AADRO

Registrul de control al convertorului A/D are configuraţia următoare:

Simbol	Poziţie	Nume şi semnificaţie
AADR0	ADCON.0	
AADR1	ADCON. 1	AADR2÷AADRO - codul binar de selecţie a intrării analogice
AADR2	ADCON .2	conectate la convertor (1/8).
ADCS	ADCON.3	Flagul de declanşare a conversiei şi de indicare a stării convertorului.
ADCI	ADCON.4	Flagul de întrerupere la terminarea conversiei. Trebuie resetat prin software, după citirea rezultatului conversiei.
ADEX	ADCON.5	Validează (pe "1") posibilitatea de a declanşa o conversie din exterior, printr-un front crescător aplicat pe linia STADC.
ADC.O	ADCON.6	Bitul 0 al rezultatului conversiei.
ADC.1	ADCON.7	Bitul 1 al rezultatului conversiei.

Un front activ pe linia STADC este detectat la sfârşitul unui ciclu maşină, iar conversia începe odată cu următorul ciclu maşină. Când este declanşată prin software, conversia începe odată cu ciclul maşină care urmează după cel al instrucţiunii de setare a bitului ADCS. Acest bit este implementat fizic cu ajutorul a două bistabile: unul poate fi setat prin program la o comandă de declanşare a conversiei (Start), celălalt poate fi numai citit, indicând starea convertorului (Status).

Următoarele două cicluri maşină sunt utilizate pentru iniţializarea logicii de control a conversiei. La sfârşitul primului ciclu este setat flagul de stare ADCS, indicând astfel o conversie în curs de desfăşurare. Eşantionarea intrării analogice selectate începe la sfârşitul celui de-al doilea ciclu şi durează încă alte 8 cicluri maşină, în acest timp tensiunea aplicată la intrare trebuie să fie relativ stabilă, viteza sa de variaţie ("slew rate") trebuind să nu depăşească 10V/ms. în final sunt parcurşi 10 paşi de aproximare succesivă, fiecare consumând 4 cicluri maşină, în total, o conversie durează 2+8+10x4 = 50 de cicli maşină, adică 50µs la fosc=12MHz.

Terminarea conversiei este semnalizată prin setarea flagului de întrerupere ADCI (ADCON.4), moment în care ADCS este resetat. ADCS nu poate fi resetat prin program,

iar ADCI nu poate fi setat. Rezultatul conversiei se află în registrul aproximațiilor succesive şi poate fi citit de către CPU din SFR ADCH (cei mai semnificativi 8 biți, ADC.9+ADC.2) şi din ADCON.7 (ADC.1) şi ADCON.6 (ADC.O).

O conversie în curs de desfăşurare nu este afectată de o nouă comandă de start. O nouă conversie poate fi comandată numai după resetarea bitului ADCI. Chiar dacă este posibil ca bitul ADCI să fie setat şi ADCS resetat printr-o singură instrucțiune (când se startează o nouă conversie pe acelaşi canal), producătorul recomandă ca mai întâi să se reseteze ADCI şi abia apoi să se seteze ADCS.

Tensiunile de referință AVref+ şi AVref- trebuie să fie cuprinse între AVdd+0,2V şi AVss-0,2V. AVref+ trebuie să fie mai mare decât AVref-, iar tensiunea de intrare VIN trebuie să fie cuprinsă între AVref+ şi AVref-. Spre exemplu, dacă gama tensiunilor de intrare VIN este cuprinsă între 2V şi 4V, atunci o rezoluție de 10 biți poate fi obținută dacă AVref+=4V şi AVref-=2V. Rezultatul conversiei poate fi calculat cu relația:

ADC =l024 x (VIN - AVref-) / (AVref+ - AVref-).

Pentru a putea face o coversie A/D este nevoie de surse de tensiune pentru asigurarea tensiunilor de referință Avref+ şi Avref-.

Semnalul care urmează a fi convertit poate să provină de la o sursă independentă sau se poate prelua de la un divizor de tensiune legat la sursele care asigură tensiunile de referință, în acest fel se poate regla tensiune de intrare nemaifiind nevoie de încă o sursă de tensiune.

Tensiunile de referință sunt setate la laborator astfel: Avref+ = 4V şi Avref- = 2V. Semnalul analogic de convertit este furnizat de o sursă de tensiune reglabilă şi afişaj digital.

12.2. Exemplu de program

Să se realizeze un program care să facă conversia unei valori analogice şi să o transmită calculatorului prin interfața serială.

```
; declararea adreselor RAM ale resurselor suplimentare
        ADCON       equ     0C5h
        ADCH equ    0C6h
        PWMPequ     0FEh
        IEN0   EQU  0A8h
        S0CON       EQU  098h
        S0BUF       EQU  099h
        ORG  8000H
        LJMP  MAIN
```

```
ORG 8090H
MAIN:
        MOV   PCON,#00h
        MOV   IEN0,#00010000b     ; invalidarea intreruperilor
        MOV   TMOD,#00100000b     ; Timerul 1 generator de tact in modul 2
        MOV   TH1,#0FDh           ; viteza de comunicare este 9600 Baud
        MOV   TL1,#0FDh           ; initializarea registrului de reincarcare a Timerului
1
        SETB  TR1                 ; pornirea portului serial
        MOV   S0CON,#01010000b    ; programare port serial in modul 1

        MOV   DPTR,#TEXT          ; in DPTR adresa de inceput a sirului TEXT
        ACALL         TRX1        ; apel rutina de transmisie mesaj
;       MOV   DPTR,#TEXT1         ; in DPTR adresa de inceput a sirului TEXT1
;       ACALL         TRX1        ; apel rutina de transmisie mesaj
ACH:                              ; inceputul rutinei de achizitie
        CLR   A                   ; initializeaza acumulatorul cu 0
        MOV   ADCON,A             ; initializeaza registrul de comanda A/D
        MOV   R1,#0H              ; adresa canal 0 in R1
        MOV   A,R1                ; A:=adresa canal 0
        ORL   A,#08H              ; bitii de control
        MOV   ADCON,A             ; start conversie
WAIT:
        MOV   A,ADCON             ; A:=registrul de stare al A/D
        JNB   ACC.4,WAIT          ; asteapta terminarea conversiei
        MOV   A,ADCH              ; citeste rezultatul conversiei
        MOV   B,#33H              ; impartire cu 51=255/5
        DIV   AB                  ; obtine cifra unitatilor
        PUSH  ACC                 ; salveaza cifra unitatilor
        MOV   A,B                 ; reface restul 1
        MOV   B,#10               ; B:=10
        MUL   AB                  ; inmultire cu 10
        MOV   R7,A                ; stocheaza octet inferior rezultat
        MOV   A,B                 ; restaureaza octet superior rezultat
        ANL   A,#01H              ; test depasire
        JNB   ACC.0,LABEL1; salt continuare daca nu e depasire
        ADD   A,#4                ; calculul partii semnificative a catului
LABEL1:
        CLR   C                   ; anuleaza CARRY
        MOV   R4,A                ; stocheaza partea semnificativa a catului
        MOV   B,#33H              ; impartire cu 51=255/5
        MOV   A,R7                ; restaureaza octet inferior rezultat
        DIV   AB                  ; calculeaza cat 1 partial
        ADD   A,R4                ; calculeaza prima zecimala
        PUSH  ACC                 ; salveaza prima zecimala
```

```
  MOV       A,B                   ; reface restul 2
        MOV  B,#10                 ; B:=10
        MUL  AB                    ; inmultire cu 10
        MOV  R7,A                  ; stocheaza octet inferior rezultat
        MOV  A,B                   ; restaureaza octet superior rezultat
        ANL  A,#01H                ; test depasire
        JNB  ACC.0,LABEL2; salt continuare daca nu e depasire
        ADD  A,#4                  ; calculul partii semnificative a catului
LABEL2: CLR  C                     ; anuleaza CARRY
        MOV  R4,A                  ; stocheaza partea semnificativa a catului
        MOV  B,#33H                ; impartire cu 51=255/5
        MOV  A,R7                  ; restaureaza octet inferior rezultat
        DIV  AB                    ; calculul cat 2 partial
        ADD  A,R4                  ; calcul a doua zecimala
        ACALL       HEXASC         ; convertire cod ASCII
        POP  ACC                   ; reface prima zecimala
        ACALL       HEXASC         ; converteste in cod ASCII
        POP  ACC                   ; reface cifra unitati
        ACALL       HEXASC         ; converteste in cod ASCII
        AJMP ACH                   ; proces ciclic de conversie

HEXASC:                            ; rutina de conversie hexa-ascii
        ANL  A,#0FH                ; se mascheaza cei patru biti mai
                                   ; semnificativi ai acumulatorului
        JNB  ACC.3,NOADJ           ; daca bitul 3 = 0 salt la NOADJ
        JB          ACC.2,ADJ      ; daca bitul 2 = 1 salt la ADJ
        JNB  ACC.1,NOADJ           ; daca bitul 1 = 0 salt la NOADJ
ADJ:
        ADD  A,#07H                ; aduna acumulatorul cu #07H
NOADJ:
        ADD  A,#30H                ; aduna acumulatorul cu #30H
        RET                        ; revenire din rutina HEXASC

TRX:
        MOV  S0BUF,A               ; A contine codul de afisat
        JNB  TI,$
        CLR  TI
        RET

TRX1:
        CLR  A                     ; initializeaza acumulatorul cu 0
        MOVC        A,@A+DPTR      ; muta in acumulator adresa gasita
                                   ; la adresa @A+DPTR
        CJNE A,#24H,TRCAR1         ; compara continutul acumulatorului cu #24H ($)
        RET                        ; iesire din subrutina TRX1
TRCAR1:
```

```
        MOV   S0BUF,A
        JNB   TI,$
        CLR   TI
        INC   DPTR              ; adresa urmatorului caracter de scris
        SJMP  TRX1              ; pentru a scrie urmatorul caracter

TEXT:   DB ' ACHIZITIE A/D Vin Ch. : $'; text prezentare
END                     ; sfarsitul programului
```

Nu se pot face şi programe mai simple cu convertorul analog digital?

Ba da, de exemplu se pot citi de la convertor doar cei 8 biţi mai semnificativi. De asemenea programul poate să afişeze direct pe LED-uri valoarea celor 8 biţi mai semnificativi, rezultatul conversiei fiind vizibil şi spectaculos. Astfel programul devine mult mai simplu.

Va fi cerut la test un program de lucru cu convertorul analog digital?

Nu, în această şedinţă de laborator cei care au reuşit programarea portului paralel, serial şi a timerului pot să evolueze şi să înveţe programarea convertorului analog digital. Studenţii care mai au probleme rămase de la lucrările anterioare le pot lămuri în această şedinţă.

Rezumat

În această lucrare se studiază şi se programează convertorul analog digital. Schema de principiu, explicarea funcţionării şi descrierea programării ajută studenţii să conceapă propriul program. Exemplul de program din lucrare combină cunoştinţele noi despre convertorul analog digital cu cele din şedinţa precedentă şi arată cum se poate face o conversie şi apoi să se trimită rezultatul prin interfaţa serială. Efectul rulării acestui program este agreat de studenţi pentru că modificarea tensiunii de intrare produce modificarea imediată şi vizibilă a valorii citite prin Hyperterminal de la placa de dezvoltare.

Bibliografie

1. C. Gerigan, P. Ogrutan, *Tehnici de interfaţare*, Editura Universitatii Transilvania Brasov, 2000, pag. 201-229 online la: http://vega.unitbv.ro/~ogrutan/ti/cap10.pdf
2. http://www.fsinc.com/devtools/index.htm
3. http://www.nxp.com/documents/data_sheet/80C552_83C552.pdf
4. http://www.keil.com/dd/docs/datashts/intel/ism51.pdf