

7.

Produse industriale specifice conducerii sistemelor cu ajutorul calculatorului. Interfața de proces ADA1110

În acest capitol este prezentată documentația de firmă, pentru un produs industrial, și anume interfața de proces cu un calculator compatibil PC, ADA1110, care prin caracteristicile sale și sfera largă de aplicații dezvoltabile se încadrează în clasa produselor universale.

7.1 Compunere generală și caracteristici

Schema bloc principală este prezentată în *figura 7.1*, observându-se reunirea pe aceeași placă a circuitelor specializate pentru diferite aplicații analizate în capitolele anterioare: conversia analog-numerică, conversia numeric-analogică, prezența unui timer (numărător-temporizator) cu aplicații în gestionarea factorului timp și existența unor intrări/ieșiri logice; tocmai această reunire a atâtor facilități pe o singură placă îi conferă universalitate acesteia.

Se prezintă în continuare principalele caracteristici și blocuri componente:

a) Blocul intrărilor analogice

- sunt prezente 16 canale pentru intrări analogice, (notate cu AIN1..AIN16) tensiunile de intrare putând fi selectate în 3 domenii: +-10V, +-5V și [0,10]V; selecția uneia dintre cele 3 variante se efectuează prin intermediul a două jumpere poziționabile pe switchul P7;

- de asemenea, există posibilitatea adăugării unor rezistențe suplimentare, în scopul realizării unei amplificări dorite după multiplexare, în cazul prezenței la intrare a unor tensiuni diferite de +-5V, +-10V sau [0,10]V.

- intrările sunt multiplexate pentru atacarea circuitului de eșantionare-numărare și amplificare. (S/H sample and hold amplifier), aceste operații (multiplexarea și eșantionarea având loc prin control soft;

- impedanța de intrare a fiecărui canal este de $> 10\text{MHz}$;

- protecția la intrare este asigurată pentru tensiuni de până la +-35V c.c;

- CAN este de tipul AD574, lucrând pe principiul aproximațiilor succesive, având caracteristicile: rezoluție de 12 biți (semnificând 2,44mV pentru un domeniu de 10V sau 4,88mV pentru un domeniu de 20V), liniaritate +-LSB, viteză de conversie 20us, frecvența de trecere – rata de achiziție (throughput) fiind de până la 40kHz, iar timpul de stabilizare (settling time) de 1us;

- se recomandă legarea canalelor de intrare AIN1..AIN16 neutilizate la masă, pentru a nu fi afectată precizia rezultatelor;

- semnalul aplicat la intrare se va conecta la uina dintre intrările AIN1..AIN16 în timp ce masa acestuia se va conecta la pinul ANALOG GND;

- tabele de corespondență în cadrul conversiilor A<->N sunt similare cu cele care vor fi prezentate la blocul CNA.

b) Blocul de conversie numeric-analogică

- există 2 ieșiri analogice, notate cu AOUT1,2;

- sunt utilizate CNA de tipul 7237, lucrând pe 12 biți, generând la ieșire 4 game de tensiuni (+-5V, +-10V, 0.5V, 0.10V), deci unipolare sau bipolare, aceste opțiuni fiind selectabile prin 2 jumpere de pe switchul P4 pentru AOUT1 și respectiv P5 pentru AOUT2;

- precizia și neliniaritatea sunt corespunzătoare la +-1 bit, iar eroarea de 0 de ½ bit;

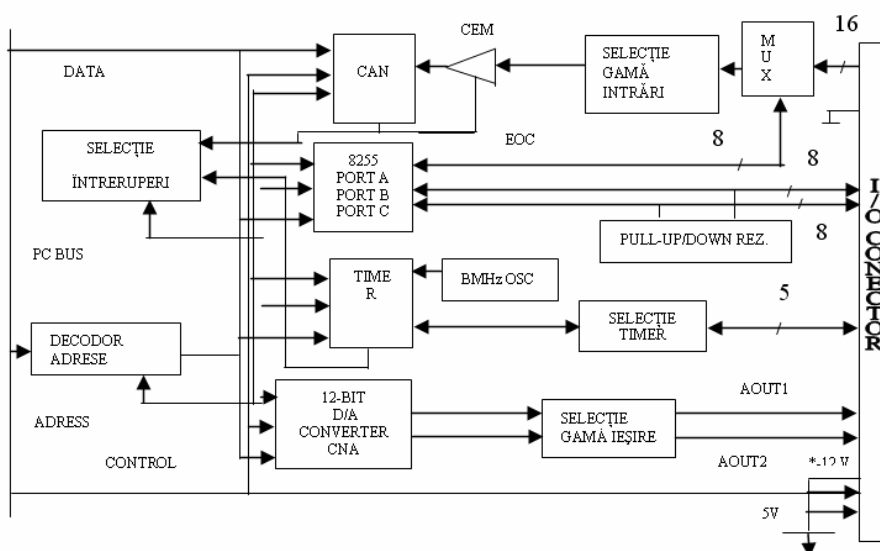


Figura 7.1 Scema-bloc a interfeței de proces ADA1110

- timpul de stabilizare (settling time) este 10 μ s;
- izolația canal-canal este de 84dB;
- curentul de ieșire maxim este 10mA;
- diafonia (crosstalk) 90dB;
- ambele canale pot fi curățate (cleared) de informațiile existente sub control soft, acesta fiind o facilitate importantă în operațiile de recalibrare sau generare de unde în formă de dinte de fierăstrău (sawtooth waveforms);
- semnalele de ieșire se obțin între pinii AOUT1,2 și ANALOG GND;
- tabela de corespondențe în cazul conversiilor pe 12 biți:

Nr. biți	[0,5] V	[0,10] V	[-5,5] V	[-10,10] V
1 LSB	1.2207mV	2.4414mV	2.4414mV	4.8828mV
4095=111111111111	4.9988	9.9976	4.9976	9.9951
2048=110000000000	2.500	5.000	0	0
1024=010000000000	1.250	2.5	-2.5	-5
.....				
1=000000000001	0.00122	0.00244	-0.00499	-0.0099
0=000000000000	0	0	-5	-10

- tabela de corespondențe A \leftrightarrow N în cazul conversiilor pe 8 biți:

Nr. biți	[0,10] V	[-10,10] V	[-5,5] V
1 LSB	39.063mV	78.126mV	39.063mV
1111 1111	9.9219	9.9219	4.9609
1000 0000	5.	0.	0.
.....			
0000 0000	0.	-10.	-5.

c) Blocul intrărilor/ieșirilor logice

- sunt disponibile 24 de linii I/O compatibile TTL/CMOS generate cu ajutorul circuitului 8255 (CMOS 82C55);

- 16 dintre acestea sunt disponibile la conectorul de ieșire, 8 sunt disponibile de pe placă;

- nivele de tensiune sunt următoarele:

- High output min 4,2V
- Low output max 0,45V
- High input min 2,2V, max 5,5V
- Low input min -0,3V,

AIN1	①	②	AIN9
AIN2	③	④	AIN10
AIN3	⑤	⑥	AIN11
AIN4	⑦	⑧	AIN12
AIN5	⑨	⑩	AIN13
AIN6	⑪	⑫	AIN14
AIN7	⑬	⑭	AIN15
AIN8	⑮	⑯	AIN16
AOUT1	⑰	⑱	ANALOG GND
AOUT2	⑲	⑳	ANALOG GND
ANALOG GND	㉑	㉒	ANALOG GND
PA7	㉓	㉔	PC7
PA6	㉕	㉖	PC6
PA5	㉗	㉘	PC5
PA4	㉙	㉚	PC4
PA3	㉛	㉜	PC3
PA2	㉝	㉞	PC2
PA1	㉟	㊱	PC1
PA0	㊲	㊳	PC0
EXT CLK 0	㊴	㊵	T/C OUT 0
EXT GATE 0	㊶	㊷	T/C OUT 1
EXT CLK 1	㊸	㊹	T/C OUT 2
EXT CLK 2	㊺	㊻	EXT GATE 1/2
+12 VOLTS	㊼	㊽	+5 VOLTS
-12 VOLTS	㊾	㊿	DIGITAL GND

d) Semnale (pini) auxiliari

- există 2 mase (grounded) analogică și digitală;
- alimentările necesare sunt +5V (68mA), +12V (20mA), -12V (28mA);

e) Semnificația pinilor

Este utilizat un conector cu 50 de pini plasați ca în figura de mai sus.

f) Blocul circuitelor numărătoare/temporizatoare

Există un circuit timer 8254 (CMOS82C54) programabil, cu 3 canale pe 16 biți, făcând posibilă dezvoltarea unor aplicații foarte variate.

- numărare cu incremente de până la 125ns, în format binar sau BCD;
- condiționat de rezistența unui impuls de ceas de 8MHz (extern sau intern al plăcii uP), poate fi folosit la generarea unor întreruperi hard sau unor semnale de strobe, de asemenea ca numărător de evenimente sau generator de impulsuri;
- fiecare canal poate fi legat (jumpered) la canalele de întrerupere ale PC sau poate fi legat în cascadă (cascaded) la un alt canal pentru a genera temporizări mai mari;
- se utilizează masa DIGITAL GND;

7.2 Inițializarea interfeței (BOARD SETTINGS)

În acest scop sunt prezente mai multe *switch*-uri astfel:

P3 - permite selecția sursei de 8MHz pentru cele 3 canale ale circuitului timer/numărător 8354; switchul permite opțiunea a 3 legături (câte una pentru ceasul fiecărui canal) selectabile dintre următoarele 8:

CLK 0 - OSC conectare la ceasul intern uP;

CLK 0 - ECO conectare la ceas extern (pin 39);

CLK 1 - OTO conectarea lui CLK 1 la ieșirea OUT 0 (deci la ieșirea canalului 0), permițând cascada canalelor 0 și 1;

CLK 1 - OSC ceas intern;

CLK 1 - EC1 ceas extern (pin 43);

CLK 2 - OT1 cascada canalelor 1 și 2;

CLK 2 - OSC ceas intern;

CLK 2 - EC2 ceas extern (pin 44);

P4 și **P5** permit selectarea gamei și polarității tensiunilor de ieșire pentru canalele AOUT1 și respectiv, AOUT2; se poziționează 2 jumperi, care ocupă deci 2 poziții dintr-un total de 4, existând următoarele posibilități:

	5	+5	X1	X2
[-5, +5] V	OFF	ON	ON	OFF
[0, +5] V	ON	OFF	ON	OFF
[-10, +10] V	OFF	ON	OFF	ON
[0, +10] V	ON	OFF	OFF	ON

P7 permite selecția gamei și polarității, tensiunilor de intrare; sunt utilizați 2 jumperi care pot ocupa 2 poziții din cele 4 posibile generând următoarele posibilități:

	20V	10V	+	+-
[-5, +5] V	OFF	ON	OFF	ON
[-10, +10] V	ON	OFF	OFF	ON
[- 0, +10] V	OFF	ON	ON	OFF

P8 permite selecția conectării semnalelor GATE ale canalelor 1 și 2 ale timerului; sunt posibile următoarele conexiuni (se alege câte una pentru fiecare din cele 2 semnale GATE):

GT1 - +5V

GT1 - EXT

GT2 - +5V

GT2 - EXT

P9 permite selectarea numai a uneia dintre sursele de întreruperi dintre 5 posibile (PC0, PC3 – de la 8255, OUT0, OUT1, OUT2 – ieșiri ale canalelor lui 8254) și de asemenea, selectarea canalului de întreruperi unde această sursă își va plasa cererea (IRQ2,3,4,5,6,7); dacă nu se dorește conectarea, jumperul de selecție al întreruperii IRQ se plasează paralel cu linia IRQ2..IRQ7.

P10 permite conectarea semnalului EOC (end of conversion) generat de CAN la sfârșitul conversiei, la unul dintre canalele pentru cererile de întreruperi IRQ2..IRQ7; dacă nu se dorește această conectare, atunci jumperul se va plasa paralel cu linia IRQ2..IRQ7.

S1 permite selecția adresei de bază, în raport cu care se vor utiliza adresele porturilor specificate lui ADA1110; această alocare este foarte importantă, deoarece eventualele suprapuneri peste adrese ocupate de alte periferice provoacă conflicte; în funcție de cele 5 poziții posibile ale switchurilor se pot obține evident $2 \times 2 \times 2 \times 2 \times 2 = 32$ de adrese de bază situate între 200h și 3f0h.

Rezistențele pull-up/down sunt utilizate în cazul grupurilor PA și PC ale lui 8255; acestea se instalează suplimentar.

Grupul de rezistențe fixe și reglabile și o capacitate (resistor configurable gain circuitry) se instalează în scopul efectuării unor CAN la care domeniul tensiunilor de intrare este altul decât cel standard. (+5, +10, 0.5V)

7.3 Harta adreselor porturilor I/O ale ADA1110

Adresele se alocă în raport cu adresa de bază (BA) având următoarele semnificații:

- BA+0 citire – se citesc biții portului PA_8255;
scriere – programare PA;
- BA+1 citire – se citește PB_8255;
scriere – se selectează unul dintre cele 16 canale active pentru CAN: AIN1..AIN16; se utilizează numai primii 4 biți, care realizează o selecție liniară: 0000 – canal 0
0001 – canal 1
...
- BA+2 identic cu BA+0 dar pentru portul PC;
- BA+3 citire – rezervat;
scriere – se configurează 8255;
- BA+4 citire – se citește valoarea existentă în canalul 0_8254;
scriere – înscrierea unei valori în canalul 0_8254; numărarea începe îndată ce numărătorul este încărcat;
- BA+5 identic cu BA+4 dar pentru canalul 1;
- BA+6 identic cu BA+5 dar pentru canalul 2;
- BA+7 citire – rezervat;
scriere – se programează timerul conform desenului 4.1;
- BA+8 citire – se citește MSB rezultat al conversiei AN (în cazul conversiei pe 8 biți aici se regăsește rezultatul, iar în cazul celei pe 12 biți aici se citesc biții de la 4 la 11);
scriere – declanșează conversia AN pe 12 biți (valoarea înscrisă nu are importanță);
- BA+9 citire – se citesc biții 0-3 în cazul unei conversii AN pe 12 biți;
Scriere – declanșează conversia AN pe 8 biți; (valoarea înscrisă nu are importanță);
- BA+10 citire – bitul 0 = EOC (End Of Convert) egal cu 0 precizează terminarea conversiei;

scrierea – declanșează o conversie simultană în ambele CNA1,2; (valoarea înscrisă nu are importanță);
 BA+11 rezervat;
 BA+12 citire – rezervat;
 scriere – LSB pentru CNA1;
 BA+13 identic cu BA+12 dar se referă la biții 8-11 ai conversiei NA, biți care se plasează pe primele 4 poziții;
 BA+14 identic cu BA+12 dar se referă la CNA2;
 BA+15 identic cu BA+13 dar se referă la CNA2;

7.4 Programarea ADA1110

În acest scop se pot utiliza tehnicile generale expuse în capitolul 2, fiind posibilă utilizarea diferitelor limbaje de nivel înalt și a celui de asamblare; fiecare dintre acestea are instrucțiuni proprii pentru scrierea și citirea porturilor, astfel:

C	data=inportb(adresa)	outportb(adresa,data)
Pascal	data:=port(adresa)	port(adresa):=data
Basic	data=INP(adresa)	OUT adresa,data
Asamblare	in al,dx	

Deși diferitele compilatoare acceptă lucrul cu porturi pe 8 sau 16 biți, în cazul utilizării lui ADA1110 se vor utiliza numai modalități de lucru pe 8 biți.

Pentru setarea, resetarea (clear) și selecția anumitor biți dintr-un port se utilizează tehnica măștilor, utilizând operatorii logici și, sau, sau exclusiv. De exemplu, se dorește repetarea (punerea pe 0 clear), a biților 2,4,6 dintr-un port; în cazul utilizării operatorului logic „și” se determină masca adecvată astfel: 2 2 6

$171=255-2-2-2,$

secvența de program fiind: v=inportb(adresa_port);
 v=v&171;
 outportb(adresa_port,v);

7.5 Tehnici de realizare a conversiilor analog-numerice

Datorită complexității acestei aplicații, în continuare se prezintă etapele necesare realizării unei conversii AN:

-se programează interfața 8255, portul B fiind de ieșire iar grupul B (portul B+portul C low) în modul 0; în acest scop cuvântul de comandă, care se înscrie la adresa BA+3 este:

1****00*, valorile marcate cu * ne reprezentând importanță

-se selectează canalul dorit (AIN1.....AIN16), înscrierea la adresa BA+1 a unui octet adecvat; sunt utilizați numai primii 4 biți pentru a selecta unul dintre cele 16 canale :(de exemplu, pentru selectarea canalului 12 se transmite octetul ****1011);

-se declanșează conversia prin înscrierea unei valori (valoarea nu contează) la adresa BA+9 în cazul conversiilor pe 12 biți;

-se urmărește (monitoring) evoluția conversiei prin citirea (înterogarea) bitului 0 (EOC=End Of Conversion) al octetului de stare de la adresa BA+9; tranziția de la 0 la 1 a bitului menționat specifică terminarea conversiei;

-la 20 us de la terminarea conversiei, se citesc succesiv LSB și MSB de la adresele BA+9 și BA+8;

-se formează rezultatul conversiei cu ajutorul relației: rezultat=MSB*16+MSB/16;

-se interpretează rezultatele obținute ținând cont de domeniul de valori și de polaritatea tensiunii de intrare, de exemplu, citirea valorii 1024 în cazul unei conversii pe 12 biți va fi interpretată astfel:

1. În cazul domeniului [0,10] V rezultatul este $1024 * 2.4414 \text{ mV/bit} = 2.499 \text{ V}$;
2. În cazul domeniului [-5, +5] V rezultatul va fi (1024-2048) biți $2.4414 \text{ mV/bit} = -2.499 \text{ V}$; scăderea lui 2048 reprezintă o translație spre minus cu jumătate din valoarea domeniului, în biți reprezentând 2 la puterea 11 = 2048;
3. În cazul domeniului [-10, +10] V rezultatul va fi (1024-2048)*4.88mV/bit= -4.999V;

În cazul unei conversii pe 8 biți factorii de scară sunt 1LSB=39.063mV/bit în cazul domeniilor cu extensie de 10V și 78.126mV/bit în cazul domeniului [-10 , +10] V.

7.6 Exemple de programe scrise în C

Se prezintă în continuare un pachet de fișiere sursă C, care permit exploatarea interfeței ADA1110.

Fișierul ada1110.h definește prin intermediul unor directive preprocesor adresele relative ale porturilor componente:

```
#define PPI_A 0
#define PPI_B 1
#define CHANNEL_SLCT 1
#define GAIN_SLCT 1
#define PPI_C 2
#define PPI_CTRL 3
#define TIMER_A 4
#define TIMER_B 5
#define TIMER_C 6
#define TIMER_CTRL 7
#define START_CONVERSION 8
#define READ_DATA_MSB 8
#define READ_DATA_LSB 9
#define STATUS_BYTE 10
#define DAC_UPDATE 10
#define DAC1_LSB 12
#define DAC1_MSB 13
#define DAC2_LSB 14
#define DAC3_MSB 15
#define PPI_PORT_A 0
#define PPI_PORT_B 1
#define PPI_PORT_C 2
#define BIPOLAR 1
#define UNIPOLAR 0
```

Fișierul ada1110.inc conține mai multe funcții generale utilizabile în scopul diferitelor utilizări ale ADA1110.

```
#include<dos.h>
#define ENABLED 1
#define DISABLED 0
#define INPUT 1
#define OUTPUT 0
#define TRUE 1
#define FALSE 0
```

```
unsigned BaseAddress;
```

```
float VoltageRange,  
      DACSlope,  
      ConversionFactor,  
      BaseLine;  
int DACOffset;  
unsigned char ANDBits;
```

*/*Funcția InitializeBoardSettings() este utilizată pentru setarea adresei de bază și determinarea factorului de conversie*/*

```
void InitializeBoardSettings(unsigned BA, float Range, char Polarity) {  
    BaseAddress=BA;  
    VoltageRange=Range;  
    ConversionFactor=VoltageRange/4096.0;  
    If (Polarity==BIPOLAR)  
        Baseline=0;  
    Else  
        Baseline=5.0;  
}
```

*/*Funcția Function DigitalToReal() convertește o valoare întreagă într-o valoare reală corespunzând unei tensiuni analogice exprimate în volți*/*

```
float DigitalToReal(int DigitalValue) {  
    return (DigitalValue*ConversionFactor+Baseline);  
}
```

*/*Funcția ResetBoard() este utilizată pentru resetarea (reinițializarea) lui ADA1110; interfața 8255 este configurată astfel că porturile A și C sunt porturi de intrare, iar portul B este port de ieșire; de asemenea declanșează efectuarea unei conversii elementare*/*

```
void ResetBoard(void) {  
    unsigned char B;  
    outputb(BaseAddress+PPI_CTRL, 0x99);           //port B -  
output  
    outputb(BaseAddress+PPI_B, 0);                 //canalul0  
    outputb(BaseAddress+START_CONVERSION, 0);     //start  
conversion  
    while ((inportb(BaseAddress+STATUS_BYTE) & 1)==0); //asteptare  
    pana la terminarea //conversiei
```

```
        B=inportb(BaseAddress+READ_DATA_LSB);
        B=inportb(BaseAddress+READ_DATA_MSB);
    }

/*Functia SetChannel este utilizata pentru setarea bitilor canalului activ CAN
in registrul atasat; se precizeaza canalul activ*/
void SetChannel(unsigned char ChannelNumber)
{
    unsigned char B;
    B=inportb(BaseAddress+CHANNEL_SLCT); //citire octet curent
    B=B & 240; //anulare B0-B3
    B=B | (Channel-1); //setare biti
    Outportb(BaseAddress+CHANNEL_SLCT,B); //scriere octet nou
}

/*Functia StartConversion() declanseaza conversia AN*/
void StartConversion(void) {
    outportb(BaseAddress+START_CONVERSION,0);
}

/*Functia ConversionDone() returneaza TRUE(#0) daca conversia AN este
completa si FALSE daca este in desfasurare*/
char ConversionDone(void)
{
    unsigned char Status;
    Status=inportb(BaseAddress+STATUS_BYTE); //recitire stare CAN
    If ((Status & 1)==1)
        return (TRUE);
    Else
        return (FALSE);
}

/*Functia ReadData() recupereaza cei 2 octeti ai CAN si ii combina intr-o
valoare intreaga*/
int ReadData(void)
{
    int MSB,LSB;
    MSB=inportb(BaseAddress+READ_DATA_MSB) * 16;
```

```
        LSB=inportb(BaseAddress+READ_DATA_LSB) / 16;
        return (MSB+LSB-2048);
    }

/*Functia ClockMode() este utilizata pentru a stabili modul de lucru al
timerului 8254 si a selecta canalul clock ={0,1,2}*/
void ClockMode(unsigned char Clock, unsigned char Mode)
{
    unsigned char StatusByte;
    StatusByte=(Clock*64)+(Mode*2)+48;
    outportb(BaseAddress+TIMER_CTRL, StatusByte);
}

/*Functia ClockDivisor() este utilizata pentru incarcarea canalului desemnat
prin clock cu o valoare pe 2 biti, care va fi utilizata in functie de modul de
lucru ales*/
void ClockDivisor(unsigned char Clock, unsigned int Divisor)
{
    unsigned char MSB,LSB;
    unsigned int PortID;
    PortID=BaseAddress+TIMER_A+Clock;
    LSB=Divisor%256;
    MSB=Divisor/256;
    outportb(PortID, LSB);
    outportb(PortID, MSB);
}

void SetUserClock(float Rate) //programeaza cele 3 canale 8254 in modul 2
                              //incarcandu-le cu //diferite valori*/
{
    ClockMode(0, 2);
    ClockDivisor(0, 2);
    ClockMode(1, 2);
    ClockDivisor(1, (400000.0/Rate));
    ClockMode(2, 2);
    ClockDivisor(2, 10);
}
```

```

char ClockDone(unsigned char Timer) //citeste valoarea curenta a canalului
                                     precizat prinTimer //si returneaza
                                     False la valori mai mari ca 0
{
    unsigned int CounterValue;
    unsigned char LSB, MSB;
    LSB=inportb(BaseAddress+TIMER_A+Timer);
    MSB=inportb(BaseAddress+TIMER_A+Timer);
    CounterValue=(MSB*256)+LSB;
    If (CounterValue>1)
        return (False)
    Else
        return (TRUE);
}

/*Functia ReadDigitalIO() returneaza (citeste) valoarea unui port al interfetei
8255, precizat prin PPI_A, PPI_B, PPI_C*/
unsigned char ReadDigitalIO(unsigned char InputPort)
{
    return(inportb(BaseAddress+PPI_A+InputPort));
}

/* Functia WriteDigitalIO() realizeaza inscrierea in portul lui 8255 precizat de
OutPort = PPI_A, B, C a valorii v*/
void WriteDigitalIO(unsigned char OutputPort, unsigned char v)
{
    outportb(BaseAddress+PPI_A+OutputPort, v);
}

void ConfigureIOPorts(unsigned char PortA, unsigned char PortC)
//este utilizata pentru configurarea porturilor A si C; portul B ramane setat ca
port de iesire; //1=input port, 0=output port
{
    unsigned char ControlByte;
    ControlByte=128+(PortA*16)+(PortC*9); //cuvant comanda
    outportb(BaseAddress+PPI_CTRL,ControlByte);
}

```

/*Functia UpdateDAC() selecteaza domeniul de tensiuni in cazul conversiilor NA; variabilele DACSlope si DACOffset trebuie setate reprezentand factorii de conversie si de translatie pentru domeniul de iesire al tensiunilor CAN, conform tabelului in cazul conversiilor pe 12 biti*/

```
void UpdateDAC(unsigned char DAC, float Volts)
{
    int Value;
    Value=(int)(Volts+DACSlope)+DACOffset;
    outportb(BaseAddress+DAC1_LSB+(DAC-1)*2, Value % 256);
    outportb(BaseAddress+DAC1_MSB+(DAC-1)*2, Value / 256);
    outportb(BaseAddress+DAC_UPDATE,0);    //start CNA
}
```

Functia SetMUXChannel() selecteaza un canal al multiplexorului; inaintea apelarii acesteia portul C al lui 8255 trebuie configurat ca port de iesire*/

```
Void SetMUXChannel(unsigned char Channel)
{
    WriteDigitalIO(PPI_PORT_C, (Channel-1));
}
```

Programul dac.c demonstreaza utilizarea CAN din cadrul placii ADA1110; programul realizeaza scanarea (parcurgerea) domeniului de iesire in tensiuni al convertoarelor CAN cu un pas de valoare egala cu un milivolt*/

```
#include<dos.h>
#include<conio.h>
#include "ADA1110.h">
#include "ADA1110.inc">
int i, ScanStart, ScanEnd;
int AtoDConversion(unsigned char Channel)
//aceasta functie realizeaza conversia invers returnand valoarea numerica
{
    SetChannel(Channel); //selectarea canalului
    StartConversion(); //start conversie
    while (ConversionDone()==0); //asteptare terminare conversie
    return(ReadData()); //returneaza valoarea numerica
}
void ProgramTitle(char St[]) //gestiunea ecranului
{
```

```
        gotoxy(1,1); clrcol();
        cprintf("ADA1110 Sample Program");
        gotoxy(80-strlen(St),1);
        cprintf(St);
    }

void main()
{
    InitializeBoardSettings(768,10.0,BIPOLAR); //setarea adresei de
baza si domeniului /
    //de tensiuni de iesire
    clrscr();
    ProgramTitle("Digital to Analog Conversions");
    gotoxy(1,25);
    cprintf("Press any key to quit...");
    gotoxy(28,12);
    cprintf("Volts: ");
    ResetBoard();
    DACSlope=4095.0/10.0; //factor de scara
    DACOffset=2048; //factor de translatare
    ScanStart=-5000;
    ScanEnd=5000;
    i=ScanStart; //start de la valoarea ScanStart
    while(kbhit()==0)
    {
        UpdateDAC(1,i/1000.0); //trimiterea valorii la CAN
        gotoxy(35,12);
        cprintf("%6.3f",DigitalToReal(AtoDConversion(1)));
        delay(50);
        i++;
        i+=9;
        if (i>ScanEnd)
            i=ScanStart;
    }
    UpdateDAC(1,0.0); //seteaza CNA1 la 0.0 volti
    getch();
    clrscr();
}
```

Programul (fișierul) digital.c demonstrează scrierea și citirea porturilor interfeței paralele 8255.

```
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include "ADA1110.h">
#include "ADA1110.inc">
char ExitProgram, S[10], C;
unsigned char B;
void ProgramTitle(char S[]);
{
    gotoxy(1,1); clrcol();
    cprintf("ADA1110 Sample Program");
    gotoxy(80-strlen(S),1);
    cprintf(S);
}

void main(void)
{
    InitializeBoardSettings(768,10.0,BIPOLAR); //setarea adresei de
baza si domeniului /
//de tensiuni
    clrscr();
    ProgramTitle("Digital I/O");
    gotoxy(1,24);
    cprintf("Enter value to output (ESCAPE to exit: ");
    gotoxy(31,11);
    cprintf("IN: ");
    ResetBoard(); //resetarea placii
    ConfigureIOPorts(INPUT,OUTPUT); //configurare A-input, port
C-output
    ExitProgram=0;
    while(!ExitProgram)
    {
        gotoxy(35,11);
        cprintf("%6d",ReadDigitalIO(0)); //citire port A
        if (kbhit())
```

```

        {
            C=getch();
            If (c==27)
                ExitProgram=1;
            Else
            {
                gotoxy(42,24);
                putch(ch);
                S[0]=C;
                gets(&S[1]);
                sscanf(S,"%d",&B);
                gotoxy(42,24);
                clrhol();
                WriteDigitalIO(2,B); //scriere valoare in
portul C
            }
        }
    }
    ConfigureIOPorts(INPUT,INPUT); //Port A=INPUT, port
C=INPUT
    clrscr();
}

```

Programul softtrig.c realizează achiziția de date utilizând „triggerare”
soft:

```

#include<dos.h>
#include<conio.h>
#include "ADA1110.h"
#include "ADA1110.inc"

void ProgramTitle(char St[])
{
    gotoxy(1,1);
    clrhol();
    cprintf("ADA1110 Sample Program");
    gotoxy(80-strlen(St),1);
    cprintf(St);
}

```

```
void main()
{
    InitializeBoardSettings(768, 10.0, BIPOLAR); //setarea adresei de
                                                baza si domeniul de
                                                //tensiuni

    clrscr();

}
```

Programul timer.c demonstrează programarea circuitului timer/numărător 8254 aflat pe placa ADA1110; programul utilizează tehnica pooling, pentru depistarea sfârșitului unui ciclu (de numărare,...) al lui 8255.

```
#include<dos.h>
#include<conio.h>
#include "ADA1110.h"
#include "ADA1110.inc"
char C;
```

Programul intrpts.c demonstrează utilizarea timerului 8254 în scopul generării întreruperilor transmise la controlerul de întreruperi 8259.

```
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include "ADA1110.h"
#include "ADA1110.inc"
unsigned char OLDIMRMask; //variabila pentru memorarea IMR
unsigned char IRQ;
int ADValue; //valoare citita de la CAN
void interrupt(*OldINT)(void);

void ClearBitIMR(unsigned char Bit) //aceasta functie reseteaza (clear)
                                    bitul IMR al 8259 , //pentru a face
                                    linia omoloaga disponibila in a
                                    accepta //intreruperea

{
    unsigned char Twos[8]={1,2,4,8,16,32,64,128};
    unsigned char OldIMR, IMR;
```

```

        OldIMR=inportb(0x21);           //citire valoare curenta IMR
        IMR=OldIMR & (255-Twos[Bit]);   //clear bit dorit
        outportb(0x21,IMR);           //transmiterea noului IMR
    }

void SetBitInIMR(unsigned char Bit) //functia este asemanatoare cu
ClearBitInIMR, cu deosebirea ca aceasta seteaza (pune pe 1) bitul dorit al
IMR
{
    unsigned char Twos[8]={1,2,4,8,16,32,64,128};
    unsigned char OldIMR, IMR;
    OldIMR=inportb(0x21);
    IMR=OldIMR | Twos[Bit];
    outportb(0x21, IMR);
}

void VectorInterrupt(int InterruptNumber, void interrupt(*ISR)())
//este utilizata pentru a asigna un nou vector pentru intreruperea respectiva
{
    disable();
    setvect(InterruptNumber,ISR);
    enable();
}

void interrupt NewISR(void)
//aceasta functie este apelata cand apare o cerere de intrerupere la liniile IRQ;
este interzisa //apelarea functiilor DOS sau BIOS in cadrul functiei de tratare a
intreruperii pentru a nu crea //evenimente imprezibile
{
    StartConversion(); //start conversie
    while (ConversionDone()==); //asteptare pana la terminarea
conversiei
    ADValue=ReadData(); //transfer data de la CAN
    outportb(0x20, 0x20); //transmitere la 8259 a mesajului de sfarsit
intrerupere prin //inscriere 20h la portul
adresei 20h
}

```

```
void ConfigureIRQ(unsigned char IRQ)
{
    OldIMRMask=inportb(0x21); //salvare IMR original
    OldINT=getvect(IRQ+8);    //salvare vector initial
    VectorInterrupt(IRQ+8,NewISR); //setarea noului vector
}

void RestoreStartupIRQ(void) //restaureaza IMR initiala salvata de functia
                             //anterioara, care se va //apela intotdeauna
                             //inaintea apelului acestei functii
{
    disable();
    outportb(0x21,OldIMRMask);
    VectorInterrupt(IRQ+8,OldINT);
    enable();
}

void ProgramTitle(char St[80])
{
    gotoxy(1,1);
    clreol();
    cprintf(„ADA1110 Sample Program”);
    gotoxy(80-strlen(St),1);
    cprintf(St);
}

void main()
{
    IRQ=S;
    InitializeBoardSettings(760,10.0,BIPOLAR); //setare adresa de
                                                //baza si domeniu de
                                                //tensiuni

    clrscr();
    ProgramTitle(„Interrupt driven sampling.”);
    gotoxy(1,25);
    cprintf(„press any key to quit...”);
    ResetBoard(); //reset placa
    ConfigureIRQ(IRQ); //setup IRQ
}
```

