L11. PROIECTAREA PERIFERICELOR PE BUS-ul USB

Lucrarea de faţă îşi propune descrierea etapelor şi posibilităţilor de proiectare a unui dispozitiv periferic care să comunice cu PC-ul pe bus-ul USB. Aplicaţia realizează un dispozitiv USB, pe baza cip-ului FT232BM şi microcontrolerului ATtiny13, care efectuează măsurarea temperaturii mediului ambient printr-un senzor de temperatură LM35 şi comunică datele PC-ului prin USB.

Dispozitivul realizat foloseşte driver-ul furnizat de producătorul Future Technologies şi comunică cu PC-ul printr-un port serial virtual (*VCP*). Pentru afişarea datelor primite pe portul virtual se poate folosi orice aplicaţie capabilă să lucreze cu un port serial (*CVAVR-Terminal, Terminal, Hyperterminal etc.*) configurată la parametrii adecvaţi.

1. Metode de implementare a perifericelor pe USB

Majoritatea perifericelor se conectau prin porturile seriale şi paralele clasice, până recent cand au fost înlocuite de USB. Apariţia unei noi interfete în arhitectura PC-ului, deci şi a unui nou standard (însoţit de un protocol de comunicaţie între host/gazda PC şi periferic) presupun metode şi tehnologii noi de proiectare a perifericelor.

1.1 Implementarea perifericelor folosind controlere USB

In cazul utilizării controlerelor embedded, alegerea acestora depinde de funcţiile pe care perifericul trebuie să le execute, de cost, de dificultatea implementării, de uneltele disponibile programării, calitatea driver-elor pentru host şi de multe ori de exemplele disponibile. Unele controlere sunt înzestrate cu un CPU pe cip, în timp ce altele recurg la unul extern care comunică cu sistemul USB.

Toate controllerele beneficiază de porturi de intrare sau ieşire, buffere şi regiştri; cele cu CPU au şi chip-uri de memorie unde este executat codul. Unele dintre ele au sarcini mai complexe de efectuat decât accesarea regiştrilor, cum ar fi: gestionarea şi recuperarea descriptorilor, procesarea de date, asigurarea că pachetele de tip 'handshake' sunt trimise, etc. ?

1.2 Adaptarea perifericelor seriale clasice la bus-ul USB

Magistrala USB înlocuieşte majoritatea (dacă nu toate) interfetele vechi, odinioară uzuale pentru PC. O posibilă strategie, tentantă pentru proiectanţi, este aceea a unei abordări rapide pentru un proiect existent pe port serial prin utilizarea unei extensii. Această abordare presupune un convertor USB-serial, între interfaţa serială a controlerului embedded şi conectorul USB din PC.

Asemenea dispozitive au adesea drivere dedicate pentru diferitele sisteme de operare, oferind o emulare completă a portului serial standard.

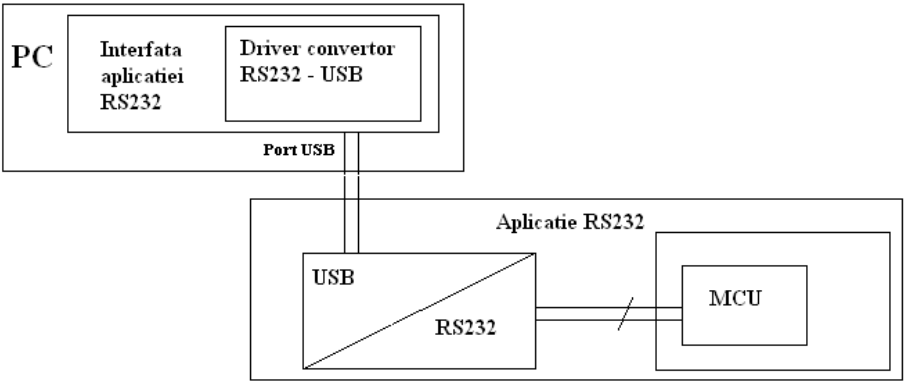


Fig. 1 Convertirea unei aplicaţii seriale RS232 la o aplicaţie seriala USB

Această abordare implică însă ca cele mai importante beneficii aduse de USB să fie pierdute. De exemplu, USB permite o viteză de transfer a datelor de până la 100 de ori mai mare decât viteza portului serial tipic (12Mbit pe secundă faţă de 115kbit pe secundă); astfel, UART devine principalul factor de limitare a transferului de date. De asemenea, natura inflexibilă a celor mai multe dispozitive de interfaţare USB-UART nu permite un management eficient al puterii, cerere prioritară pentru multe proiecte de tip embedded.

2. Proiectarea perifericelor USB cu circuite de emulare a unui port serial

Una dintre cele mai folosite metode de implementare a standardului USB încearcă să împace inovaţiile şi versatilitatea adusă de acesta cu inerţia de proiectare a perifericelor având ca interfaţă standardul RS232. Este o soluţie de compromis: faptul că portul serial clasic dispare din arhitectura multor sisteme de calcul este o indicaţie certă a dezvoltării sistemelor de comunicaţie mult mai flexibile şi de viteză mare (precum USB sau IEEE1394). Structura unui sistem de emulare a unui port serial este similară cu cea prezentată anterior; deosebirea constă în viteza de comunicaţie între PC şi periferic, viteză care depăşeşte cu mult limitele impuse de interfaţa RS232.

Pentru a întelege modul de funcţionare se va urmări schema bloc a aplicaţiei (Fig. 2) precum şi schema electrică a montajului (Fig. 3):

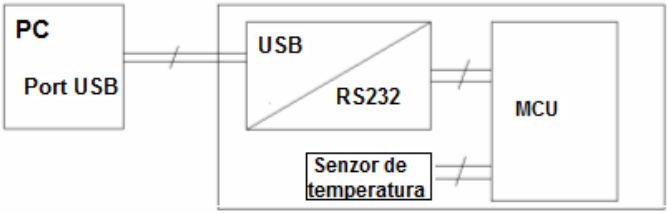


Fig. 2. Schema bloc a montajului practic

Printr-un **conector USB**, de tip B (pentru dispozitivele *slave*), se realizează legătura între PC, sau *host* şi aplicaţie. Liniile de date sunt conectate la intrările USBDM şi USBDP ale chipului FTD232BM, respectiv pinii 7 şi 8.

2.1. Componenta principală în cadrul schemei este circuitul din familia **FT232BM** fabricat de FTDI Ltd (*www.ftdichip.com*), utilizat pentru a realiza conversia USB – serial RS232. Interfaţa serială lucrează la nivele TTL, fiind compatibilă cu RS232, RS485 şi RS422 în combinaţie cu un transceiver serial (MAX232 sau similar). Rata de transfer poate ajunge la 1 Mbit/s pentru RS232 şi la 3 Mbit/s pentru RS422/RS485. Schema bloc şi funcţionarea circuitului FT232BM sunt prezentate în Anexa1. Compania producătoare pune la dispoziţia celor care doresc implementarea de proiecte cu bus-ul USB două tipuri de drivere:

(1) ***VCP*** (Virtual COM Port), ce emulează un port serial; utilizatorul va accesa perifericul conectat la PC similar unui periferic conectat la portul serial,

(2) *Driver-ele* ***D2XX*** *Direct* ce permit aplicaţiilor să funcţioneze cu circuitele FT232 şi FT245 cu ajutorul unei biblioteci Windows *DLL*.

2.2. Microcontrolerul **ATtiny13** (Atmel) are rol de procesare a informaţiilor oferite de senzorul de temperatură LM35. Legătura circuitului FT232BM cu MCU ATtiny13 se face doar prin 2 pini: RX (pin PB1 al MCU), conectat la pinul TX a convertorului FT232BM; un al doilea pin este PB0 (TX) al MCU conectat la pinul RX al convertorului. O componenta importantă a MCU este convertorul analog-numeric (CAN) intern. Cu o rezoluţie de 10 biţi, o acurateţe de ±2 LSB, un timp de conversie de 13–260 microsecunde, CAN asigură o performanţă suficientă pentru cerinţele acestei aplicatii. Schema bloc şi caracteristicile microcontrolerului ATtiny13 sunt prezentate în Anexa 1.

Accentul în aplicaţia de faţă se pune pe comunicarea perifericului cu unitatea centrală folosind busul USB.

Pentru măsurarea temperaturii s-a optat pentru un senzor de temperatură lowcost, **LM35.** Acesta este simplu şi dpdv al utilizării, modului de funcţionare şi a prelucrării datelor furnizate. Seriile LM35 au la ieşire o tensiune liniară, proporţională cu temperatura (Fig.5). In domeniul de temperatură în care

funcţionează circuitul (între -40°C şi 110°C), nu au nevoie de calibrare externă, au o acurateţe de ±0.25 ºC la temperatura camerei şi respectiv ±0.75 ºC pentru restul plajei de valori măsurabile.

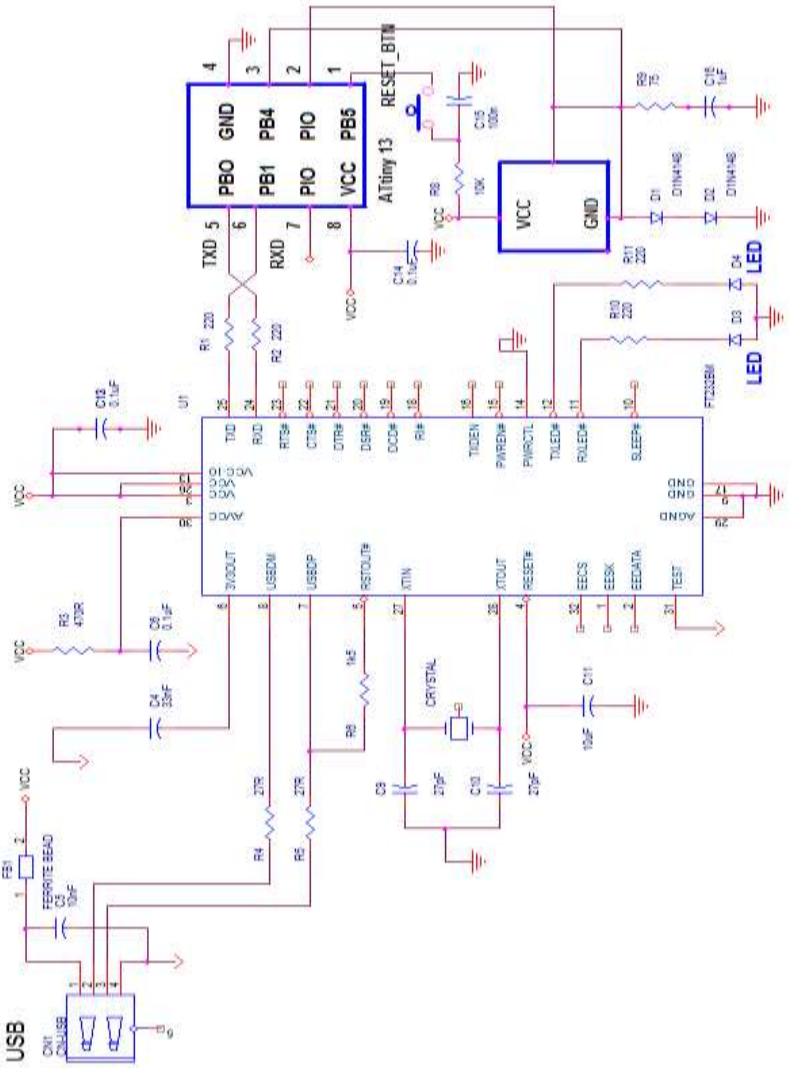
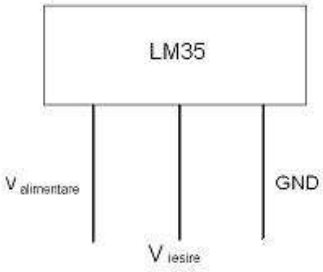


Fig. 3 Schema electrică a aplicaţiei



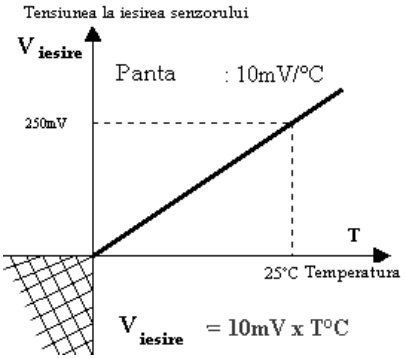


Fig. 4 Senzorul de temperatură LM35 Fig. 5 Variaţia tensiunii funcţie de temperatură

Pentru a putea măsura şi temperaturi negative cu LM35 e nevoie de o sursă de tensiune negativă. S-a optat pentru introducerea unei tensiuni de offset la terminalul GND al senzorului prin cuplarea a două diode pentru a deplasa astfel originea (Fig.6).

Căderea de tensiune cumulată pe cele două diode este de aprox. 1.2V. Fiind variabilă cu temperatura (circa 2mV/°C), nu se poate folosi căderea de tensiune pe cele două diode ca un potential de referinţă constant. Astfel, se vor realiza măsuratori diferenţiale pentru a obţine o valoare a temperaturii cât mai apropiată de realitate. Din Fig.3, tensiunea de ieşire de pe pinul **Vout** al senzorului, variabilă în mod constant cu temperatura, se conectează la pinul **PB5** al MCU, iar terminalul **GND** al senzorului la **PB4**.

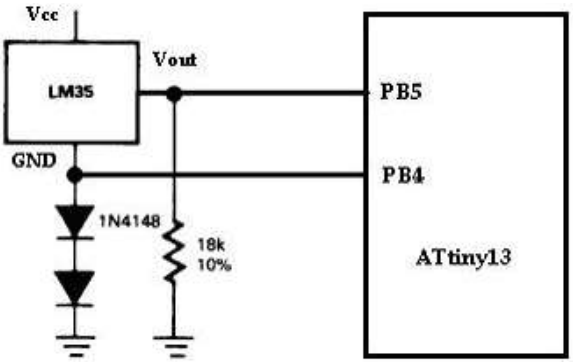


Fig. 6 Realizarea tensiunii de offset pentru posibilitatea măsurării temperaturilor negative

3. Structura firmware-ului

Microcontrolerul ATtiny13 măsoară temperatura cu senzorul de temperatură LM35 la intervale regulate de timp, după care aceasta este transmisă serial spre PC. Există posibilitatea schimbării intervalului de eşantionare a temperaturii, în acest fel implementându-se o comunicaţie bidirecţională între controler şi PC.

In Fig.7 sunt prezentate funcţiile aplicaţiei; acestea sunt activate de declanşarea întreruperilor corespunzătoare de la timer (*TIM0_COMPA*) şi a întreruperii externe (*EXT_INT0*).

Intreruperea asociată Timer-ului: ATtiny13 dispune de un numărător TIMER0 care are la rândul lui o unitate de divizare ce poate împărţi frecvenţa sistemului (9,6MHz) cu 1, 8, 64 sau 256.
Scopul aplicaţiei este de a obţine informaţii legate de temperatură la intervale regulate de timp, aceste intervale fiind de obicei de ordinul secundelor. Pentru a obţine o întârziere cu ajutorul timerului, se divizează frecvenţa procesorului cu 256 (pentru a obţine o frecvenţă de numărare cât mai mică). Intreruperea corespunzătoare timerului funcţionând în modul CTC (*Clear Timer on Compare*) se va activa la un interval de 5 ms. Pentru realizarea acestei baze de timp se înscrie în registrul OCR0A (Output Compare Register A) valoarea: 5[ms]*9.6MHz/256= 0BBh. Registrul OCR0A contine o valoare pe 8 biti care e comparata cu valoarea din contor (TCNT0). O potrivire poate fi utilizata pentru a genera o întrerupere de tip Output Compare, sau pentru a genera o formă de undă la ieşirea pinului OC0A.

Intreruperea provenită de la timer va fi folosită pentru declanşarea unei noi măsuratori de temperatură. Valoarea măsurată se depune într-un buffer circular în care se păstrează ultimele 6 valori măsurate; cea mai veche valoare măsurată este eliminată din buffer şi în plus se aplică un filtru median: se elimină valorile extreme, cea mai mică respectiv cea mai mare, şi se face media aritmetică pentru valorile rămase; valoarea rezultată, numită *temperatura_medie*, este transmisă serial către PC unde este prelucrată de interfaţa aplicaţiei.

Intreruperea externă EXT_INT0: este declanşată atunci când utilizatorul doreşte schimbarea perioadei de esantionare a temperaturii; această modificare poate surveni oricând, iar variabila *nr_sec* care temporizează întârzierile va fi actualizată. Din configuraţia pinilor pentru ATtiny13 se poate observa că pinul corespunzător întreruperii externe INT0 este PB1. Se defineşte acest pin ca RXD pentru comunicaţia serială: pe această linie se vor recepţiona datele dinspre PC. Pentru transmisia datelor către convertorul FTD232BM se foloseşte pinul 5 al microcontrolerului, PB0. Atunci când nu are loc nici un transfer de date pe RXD, pinul este în ‘1’ logic. În rutina corespunzătoare *reset*-ului (prima operaţie care se

execută odată ce este pornită aplicaţia) se setează ca INT0 să se declanşeze pe front descrescător.

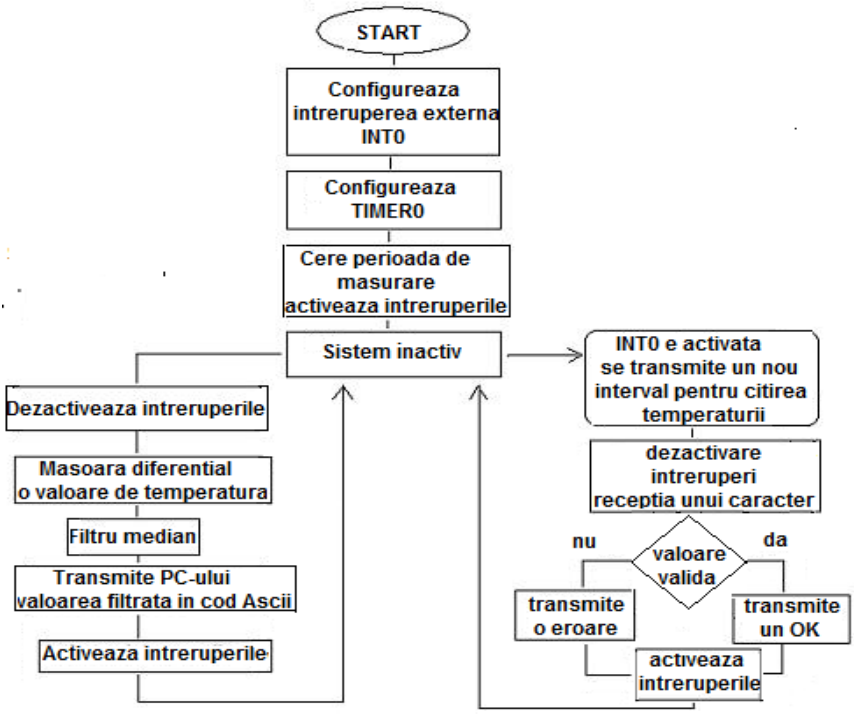


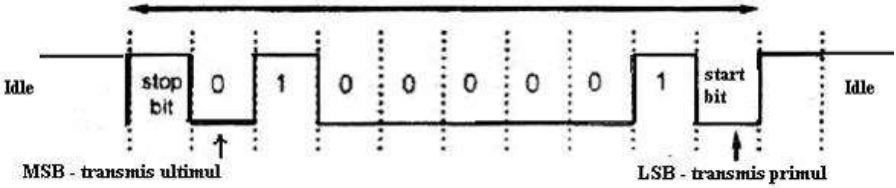
Fig. 7. Organigrama aplicaţiei

Declanşarea întreruperii externe **EXT_INT0,** coincide cu apariţia bitului de *start;* odată programul intrat în rutina care tratează întreruperea externă, se vor recepţiona toţi biţii; apoi, se apelează o rutină care va prelua date de pe RXD, bit cu bit la un debit binar de 1200 bps. Dacă recepţia s-a efectuat în parametri normali (s-au primit 8 biţi de date şi bitul de stop este activat), se trimite utilizatorului un mesaj de confirmare: "Interval actualizat!"; altfel, se transmite "Eroare!". Octetul primit reprezintă intervalul de timp care se scurge între două citiri succesive ale senzorului, (perioada de esantionare) fiind calculat după formula: *K[s] = valoare_receptionata * 1[s]*.

Astfel, pentru o *valoare_receptionata* = 0x33, corespunzător codului ASCII al cifrei 3, vom avea masurători din 3 în 3 secunde. Secventa de cod în limbaj de asamblare pentru descrierea acestei rutine este prezentat in Anexa1.

Serializarea: Serializarea informatiei binare se face prin software, deci nu folosim un circuit UART. O a doua funcţie a timer-ului este de a genera întârzierea de bit (durata bit-ului) necesară comunicării seriale la un debit binar de 1200bps.

Legătura dintre periferic/aplicatie si PC este realizată fizic prin portul USB, respectându-se protocolul **USB,** iar la nivel vizibil (de realizare a interfeţei si a softului implementat pentru MCU) se respectă specificaţiile interfeţei de comunicare serială **RS232**. Transmisia şi respectiv recepţia datelor se realizează prin şiruri de biţi, astfel: un bit de start, 8 biţi de date, 1 bit de stop. Viteza de transfer a biţilor trebuie să fie aceeaşi atât pentru PC cât şi pentru MCU (stabilită la un debit binar de 1200 bps).



Intârzierea de bit este raportul dintre viteza procesorului şi debitul binar:

delay=9.6MHz/1200bps=125

Aşadar trebuie introdusă în program o buclă care să introducă în transmisia sau recepţia unui bit o întârziere de 125 de cicluri procesor/bit.

Transmisia unui caracter: Pinul alocat pentru transmisie serială, TX, este **PB0**. Pentru ca octetul să fie reconstituit cu succes de către aplicaţie (PC), octetul se transmite bit cu bit începând cu LSB. După fiecare bit care este trimis, octetul este deplasat (shift) spre dreapta; astfel, întotdeauna se transmite bitul de pe poziţia b0.

Recepţia unui caracter: In cadrul portului de I/O al ATtiny13, pinul alocat pentru recepţia datelor, RX, este **PB1**, care în acelaşi timp corspunde şi liniei pentru întreruperea externă. Trecerea în '*low*' a lui PB1 înseamnă apariţia bitului de *start*. In acelaşi timp, se declanşează întreruperea externă şi se apelează rutina care realizează recepţia unui caracter. Pentru ca biţii să se citească cât mai corect, citirea se va face la 'mijlocul' întârzierii de bit; deci, dacă la un debit binar de 1200 bps durata unui bit este de aproximativ 832 μs, după bitul de *start* întârzierea pentru citirea bitului va fi de aproximativ 400 μs, iar pentru următorii biţi se va reveni la întârzierea normală de 832 μs.

Obţinerea valorilor de temperatură

Conversia binară a unei valori analogice: CAN poate citi 4 semnale analogice pentru intrare; în aplicaţia de faţă se foloseşte ADC3 corespunzător pinului PB3 unde este conectat semnalul de ieşire al senzorului LM35. Al doilea semnal care trebuie adus în forma binară este semnalul corespunzător terminalui GND al senzorului de temperatură conectat la pinul PB4 sau ADC2.

Configurarea senzorului de temperatură LM35

Datele furnizate de senzorul LM35 sunt semnale analogice/tensiune. Terminalul **GND** al senzorului, conectat la pinul PB4 al MCU va fi la un potenţial de aprox. 1.2 V la temperatura camerei, iar în funcţie de variaţiile de temperatură va creşte sau va scădea. Terminalul **Vout** al lui LM35 va furniza o valoare proportională cu temperatura, tensiunea **Vout** va fi liniar variabilă cu temperatura cu o pantă de 10mV/°C, fiind recepţionată de MCU prin pinul **PB5**.

Pinii PB5 şi PB4 sunt în acelaşi timp canalele ADC0, respectiv ADC2 ai convertorului analog-numeric (CAN) a microcontrolerului ATtiny13. CAN este programat să furnizeze rezultatul unei conversii pe 10 biţi (rezoluţia maximă). Modul de funcţionare este Single Conversion Mode, referinţa de tensiune coincide cu tensiunea de alimentare şi este **Vref**=5V. Astfel, având o tensiune de referinţă de 5V şi rezoluţia de 10 biţi, va rezulta: LSB = 5V/1024= 4.88mV

Dat fiind că pasul de cuantizare în volţi pentru CAN este de 4.88mV, se deduce o rezoluţie în măsurarea temperaturii de 0.48°C/cuantă, fiind aproape de rezoluţia senzorului.

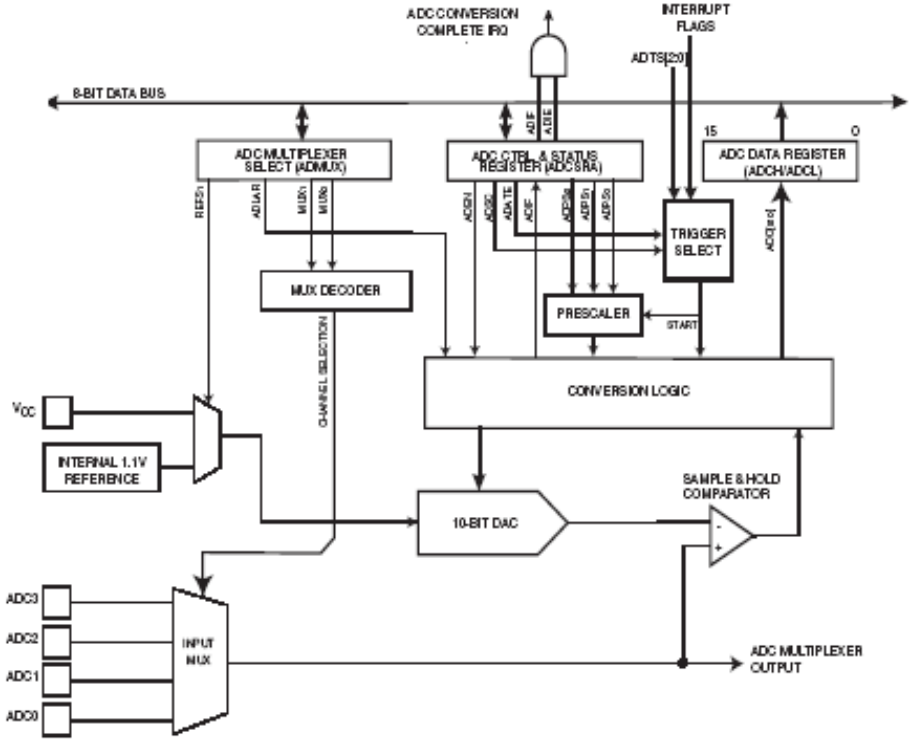


Fig. 8. Schema bloc a convertorului analog numeric din structura ATtiny13

Preluarea si prelucrarea datelor

Măsurarea temperaturii se efectuează la intervale regulate, definite de utilizator si care pot fi modificate in orice moment al rulării programului. Odată programul intrat în rutina asociată întreruperii "overflow" a timer-ului, se comandă conversia semnalului analogic de pe pinul PB3 unde este legată ieşirea senzorului. Rezultatul conversiei sunt pe doi octeţi (avem o rezoluţie de 10 biţi) şi se păstrează în doi regiştri, numiţi ***out_senzor*.** Următorul pas îl constituie măsurarea semnalului de pe pinul PB2 fata de GND, unde este legată masa senzorului; se măsoară astfel tensiunea de offset introdusă de cele două diode 1N4148 cu scopul de a permite şi măsurarea temperaturilor negative. Rezultatul este în regiştrii numiţi ***offset_senzor***. Cele două valori măsurate, *out_senzor* şi *offset_senzor* se scad pentru a obţine astfel valoarea de temperatură curenta, notată ***temperatura.*** Dacă ***offset_senzor*** > ***out_senzor*,** temperatura este negativă; rezultatele sunt pe 10 biţi şi deci sunt necesari doi octeţi pentru reprezentarea datelor.

Rezultatele scăderilor sunt depozitate în memoria SRAM a MCU, începând cu adresa 0x006C; odată ce o nouă măsurătoare este efectuată, valorile din bufferul de memorie se translatează spre stânga; astfel, prima valoare se pierde, iar noua valoare se inserează în capul listei, conform FIFO (First In, First Out).

După actualizarea bufferului, se crează o copie a acestuia, începând cu adresa 0x0060; acest nou buffer este supus următoarelor operaţii:

- se ordonează crescător şirul valorilor
- se elimină prima (cea mai mică valoare) şi ultima (cea mai mare)
- se calculează media aritmetică a valorilor rămase
- media aritmetică este stocată în regiştrii ***medie_high:medie_low***

În continuare, pentru a transmite valoarea la PC, aceasta trebuie transformată într-o valoare validă de temperatură. Se alege formatul cu 4 cifre, din care trei pentru cifre, un digit pentru punctul zecimal şi dacă este cazul la temperaturi negative se va afisa şi semnul '-'. Rezultatul CAN este :

$ADC = \frac{Vin * 1024}{Vref}$, unde: *ADC* – este rezultatul conversiei,*Vin*– valoarea analogică la intrarea convertorului, *Vref* – tensiunea de referinţă.

In cazul de faţă, *ADC* este valoarea medie, *medie_high:medie_low*, *Vref* este 5V iar *Vin* trebuie să fie marimea fizică (temperatura) cu o cifra zecimala şi două cifre pentru partea întreagă.

Unitatea de măsura pentru *Vin* este în volţi [V]; caracteristica de ieşire a senzorului este de 10mV/°C, aşadar va trebui să exprimăm *Vin* în unităţi de câte 10mV (înmulţind *Vin* cu 100); pentru a uşura calculele în program, punctul zecimal se afişeaza ulterior în cod, deci factorul de multiplicare nu va conţine şi cifrele zecimale ; rezultă o înmulţire a *Vin* cu încă 100. Formula anterioară devine:

$$Vin = \frac{ADC * Vref * 10000}{1024} * 256$$,unde: *ADC – temp_high:temp_low*, valoarea medie, *Vin* – valoarea în °C care urmează a fi transmisă,*Vref* – tensiunea de referinţă, 256 apare din codul programului când se efectuează înmulţirea cu factorul de multiplicare şi are loc o împărţire cu 2^8. Aşadar, factorul de multiplicare (*mulf*) devine:

$$mulf = \frac{Vref * 10000}{1024} * 256 = \frac{5 * 10000 * 256}{1024} = 12500$$

Emularea unui port serial (COM) virtual

Compania Future Technology Devices International Ltd (FTDI) pune la dispoziţia celor care dezvoltă proiecte având la bază cipurile FTxx driver-ele necesare comunicaţiei (în conditii bune între periferice şi sistemele de operare).

Driver-ele folosite sunt de tip VCP (Virtual Com Port), care emulează un port serial astfel încât dispozitivul de la celălalt capăt al cablului USB să poată fi accesat conform specificaţiilor RS232. Pentru instalarea driver-elor VCP, pentru un sistem de operare Windows, se descarcă driver-ele de la adresa http://www.ftdichip.com/Drivers/VCP.htm.

Pentru afişarea datelor primite pe portul serial virtual se foloseşte **Terminal** din mediul ***CodeVisionAVR***, capabil să lucreze cu un port serial, configurat la parametrii adecvaţi. Valorile de temperatură în hexa sunt greu de descifrat; astfel, temperatura va fi convertita în valori ASCII. Se va măsura temperatura şi se va trimite calculatorului în format ASCII; în acelasi timp, se încearcă schimbarea de către utilizator a intervalului între două măsurători consecutive.

Interfaţa cu utilizatorul pe PC este realizată în mediul vizual de programare Delphi 6.0. Există posibilitatea schimbării periodei de timp la care se fac măsurătorile prin editarea unei valori într-un editbox şi ulterior trimiterea acesteia pe portul serial. Există şi un memobox în care se afişează toate mesajele primite de la periferic. Interfaţa oferă şi posibilitatea schimbării setărilor legate de portul serial cum ar fi debitul binar de transfer, numărul de biţi de stop ca şi controlul parităţii.

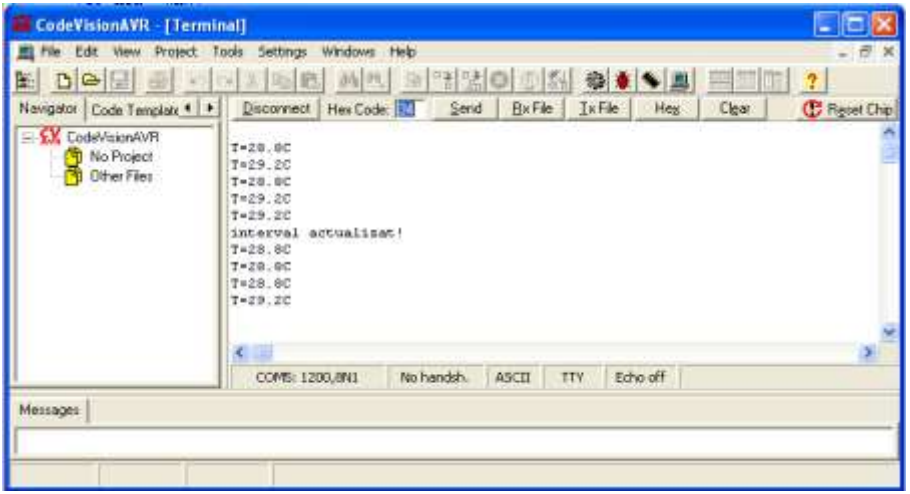


Fig. 9 Afişarea ASCII a valorilor de temperatură

Aplicaţia descrisă se doreşte a fi un început în dezvoltarea de dispozitive compatibile USB. Este oferit un model de comunicare cu PC-ul prin intermediul convertorului serie-USB FT232BM, iar de aici se pot dezvolta diferite aplicaţii.

Temă: Să se proiecteze un convertor paralel – USB folosind chipul FT245BM, pentru a conecta două PC-uri.

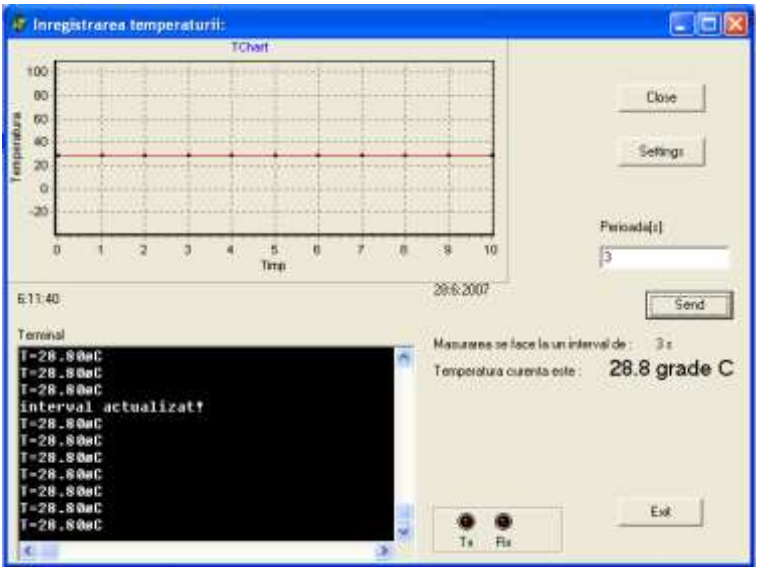


Fig. 10. Interfaţa cu utilizatorul

Mersul lucrarii:

1. Se va conecta montajul la PC si se va verifica existenta driver-ului VCP pe PC-ul gazda. In cazul lipsei driverelor necesare aplicatiei, se vor instala driver-rle VCP; pentru un sistem de operare Windows, se descarcă driver-ele de la adresa: http://www.ftdichip.com/Drivers/VCP.htm.

2. Se identifica portul serial aparut in lista porturilor LPT & COM in Device Manager.

3. Folosind CVAvr se seteaza parametrii liniei de transmisie seriala (1200bps, 8 biti/caracter, 1 bit de stop, fara paritate), se conecteaza Terminalul si se urmaresc valorile de temperatura receptionate, ca in Fig.9.

4. Se deschide fisierul .exe corespunzator aplicatiei, se seteaza parametrii liniei de comunicatie intre PC si aplicatie in cadrul interfetei (butonul *Settings*), se apasa butonul *OpenPort* pentru conectare, eventual se specifica si perioada de esantionare a temperaturii (caseta *Perioada(s)* si apoi *Send*) si se urmaresc valorile asa ca in Fig.10.

5. Se va discuta codul aplicatiei, identificand elementele cheie (intreruperi, programare timer, etc).