Tema 1: **Introducere în programarea aplicațiilor în timp real.**

§1.1. Intoducere. Noțiuni generale.
§1.2. Elementele sistemelor în timp real.
§1.3. Tipuri de sisteme în timp real.
§1.4. Clasificarea programelor.
§1.5. Definirea si caracteristicile sistemelor distribuite

§1.1. Intoducere. Noțiuni generale.

Aplicații ale sistemelor de calcul în timp - real (SCTR) sunt întâlnite în toate domeniile de activitate. Cea mai mare parte dintre aceste aplicații se refera la sisteme informatice de supraveghere si control în industrie (la nivel de utilaje, instalații, linii tehnologice, secții, întreprinderi si platforme industriale), în transport si telecomunicații, în distribuția energiei electrice, în automatizarea experimentarilor stiințifice, în activitați de management, servicii etc.

În prezent, pot fi întâlnite pretutindeni echipamente, instalații, produse de larg consum etc. care au înglobate aplicații timp - real. În [Tur99] se arata modul de împarțire a pieței comerciale de microprocesoare în anul 1999: mai puțin de 1% din lumea microprocesoarelor erau utilizate în sisteme cu întrebuințare generala. Peste 99% dintre microprocesoare erau utilizate în aplicații timp-real. Acestea sunt omniprezente, de la telefoanele celulare, pagere, cuptoare cu microunde pâna la sisteme complexe de control trafic aerian, telecomunicații, utilitați publice, conducerea proceselor industriale, multimedia etc.

Structura hardware si software-ul de baza si aplicativ prezinta trasaturi specifice pentru SCTR, ceea ce face ca aceste sisteme sa fie o categorie distincta între sistemele de prelucrare a datelor.

Într-un sistem de calcul timp - real, esențial este intervalul de timp dintre momentul introducerii datelor si momentul obținerii si interpretarii rezultatelor. Într-un sistem de prelucrare clasic, nu sunt impuse constrângeri pe fluxul introducere date – prelucrare date - obținere rezultate pe când în SCTR timpul de raspuns este esențial. Acesta este unul dintre motivele pentru care trebuie aplicate tehnici speciale pentru introducerea datelor, prelucrare si vizualizare, respectiv aplicare a rezultatelor.

Privind dintr-un cadru mai larg, SCTR fac parte din categoria sistemelor de calcul ON LINE – în care datele de intrare sunt introduse direct de la locul unde sunt produse iar rezultatele sunt transmise direct la locul de utilizare. Datele fie sunt introduse de la terminale de catre operatori umani, fie provin de la traductoare ori senzori amplasați în mediul extern. Rezultatele sunt transmise la iesire la terminale pentru vizualizare de catre operatorul uman sau la elemente de execuție.

- Def.1 Un **sistem de timp-real** este un sistem a cărui funcţionare corectă este direct influenţată de timp, sau mai exact de satisfacerea condiţiilor şi a restricţiilor de timp.
- Def. 2 Un **sistem de timp-real** este un sistem care trebuie să producă un răspuns într-un timp limitat; depăşirea acestui timp duce la degradarea calităţii serviciului sau la rezultate catastrofale.

Timpul de raspuns al unui SCTR este timpul necesar pentru a genera o informație de reacție la datele introduse.

Timpul de raspuns este o valoare absoluta care difera în funcție de cerințele aplicației timp - real:

- sisteme de recunoastere a vocii – [100ns – 10 ms];
- simulator de zbor –[1us-10us];
- simulare de procese si controlul rețelelor - [10us-100us];
- control în telemetrie si analize seismice - [100us=1ms];
- controllere pentru roboți – [1ms-10ms];
- sisteme pentru controlul proceselor si automatizari industriale – [100us-100ms];

• diagnosticare medicala automata si laboratoare automate – [10ms – 100ms];
• sisteme pentru detectare si alarmare incendiu – [100ms-1s];
• sistem rezervari locuri – [2s-3s];

În funcție de domeniul de utilizare exista mai multe tipuri de sisteme de calcul ON LINE, care înglobeaza si sistemele de calcul timp - real:

• Sisteme pentru controlul si supravegherea proceselor.

• Sisteme conversaționale – care presupun lucrul interactiv de la terminal, introducere de comenzi (date, programe etc.) si raspuns imediat la acestea.

• Sisteme tranzacționale – care sunt interactive de la terminal, dar numarul si tipul de mesaje si comenzi este limitat pentru a spori viteza de raspuns. Rezultatele se afiseaza în formate prestabilite la proiectarea aplicației. Comenzile si mesajele fixe se numesc tranzacții. Exemple: sisteme bancare, rezervare de locuri, conducere de procese în regim ghid operator.

• Sisteme timp - real încorporate (Embedded Real Time Systems). Aceste sisteme fac parte integranta dintr-un sistem tehnic ori tehnologic mai general. Exemple: un sistem pentru conducerea unui robot pe o linie de fabricație a unui produs, sistem de control al zborului în aviație, sisteme de control autovehicule.

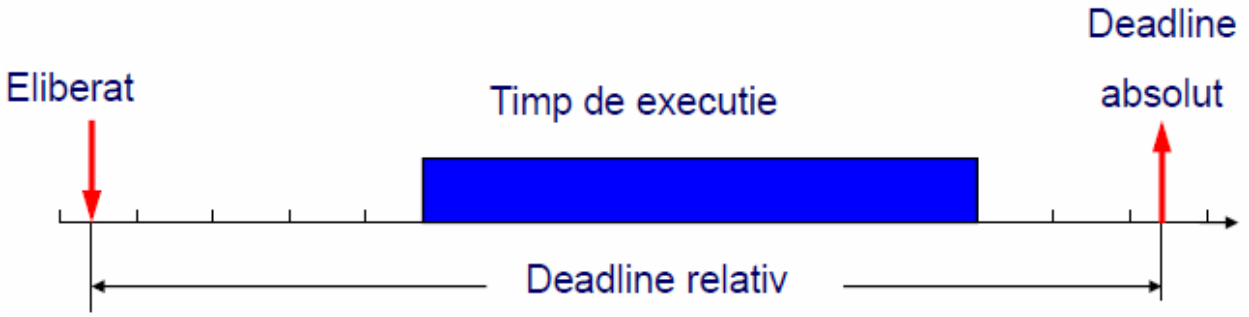
În ceea ce priveste SCTR, în literatura de specialitate exista mai multe scheme de clasificare:

• Din punct de vedere al timpului de raspuns: pot fi *Hard Real-Time systems (HRT)* sau *Soft Real-Time systems (STR)* și Sisteme de timp-real mixt

- HRT = cu constrângeri de timp dure; iesirile trebuie produse în limite de timp (deadlines) specificate, în caz contrar va aparea defect de sistem (Exemple: Sisteme de control al zborului, Sisteme de control trafic aerian, Roboti, Sisteme de control autovehicule...). Consecinte foarte grave, chiar dezastruoase pot avea loc daca termenul nu este respectat.

- SRT = cu constrângeri de timp flexibile; deadline-urile pot fi pierdute ocazional fara a se considera ca sistemul este defect (Exemple: sisteme de comunicație care utilizeaza protocoale cu „time out”, Sisteme „casier automat”, Sisteme de rezervare a locurilor, Sisteme pentru controlul proceselor proiectate sa tolereze întârzieri ...). In cazul ideal, deadline-ul trebuie respectat la performata maxima. Performanta scade in cazul ratarii termenelor limita.

- Sisteme de timp-real mixt – combina caracteristicile primelor doua sisteme.



• Din punct de vedere al deschiderii:

- SCTR proprietar, care depind de caracteristicile sistemului de operare proprietar, arhitectura hard si setul de instrucțiuni al calculatorului; costurile cu dezvoltarile soft si portarea spre alte platforme sunt foarte mari la aceste sisteme.

- SCTR deschise, care se bazeaza pe standarde industriale pentru microprocesoare, sisteme de operare timp - real, protocoale de comunicație, magistrale de interfațare; costul sistemelor este mai mic, datorita disponibilității pachetelor de programe, usurinței de integrare si utilizare, independența de platforme hard proprietar, etc.

• Din punct de vedere al arhitecturii:

- SCTR centralizate, în care procesoarele sunt localizate într-un singur nod din sistem iar timpul de comunicare între procese este neglijabil în raport cu timpul lor de execuție; un sistem multiprocesor cu memorie partajata este un exemplu de sistem centralizat.

- SCTR distribuite, în care procesoarele sunt distribuite în diverse locuri din sistem iar timpul de comunicare între procesele de pe diferite procesoare nu este neglijabil comparativ cu timpul lor de execuție; un exemplu de astfel de sistem este o rețea locala de calculatoare.
Sistemele TR pot fi ilustrate la diferite **nivele de abstractizare**.La un nivel înalt de abstractizare, un sistem TR apare ca format din **trei subsisteme** componente.

- **Subsistemul controlat** reprezinta aplicatia sau mediul care dicteaza cerintele TR.
- **Subsistemul de control** este întregul echipament de calcul, care este conectat cu mediul controlat printr-un numar de intrari si iesiri, formând interfata aplicatiei. Subsistemul de control poate cuprinde unul sau mai multe procesoare si resurse. Procesoarele si resursele sunt gestionate de un sistem software denumit *sistem de operare TR - SOTR*. În mod normal, un sistem TR are o interfata cu un operator uman,
- **subsistemul operator,** care initializeaza si monitorizeaza întregul sistem, dând anumite comenzi, mai ales în situatii exceptionale. Sistemele TR ale viitorului se doresc a fi suficient de inteligente, sigure, autonome, pentru o *executie independenta* de operatorul uman [Stankovic92].

Fiecare stimul determina anumite prelucrari în sistemul TR, rezultatele putând duce la actualizarea unor date interne, referitoare la sistemul condus sau la trimiterea unui semnal de raspuns spre exterior. Setul de pasi care se executa între receptionarea unui stimul si raspunsul sistemului la acesta, se numeste **tranzactie TR**. Conducerea unui proces impune anumite constrângeri temporale tranzactiilor TR, acestea fiind esentiale în specificarea sistemului TR.

În specificarea sistemelor TR trebuie sa se tina cont de doua **ipoteze**, estimari:

- **ipoteza de încarcare** - *load hypothesis* - se refera la rata maxima cu care poate apare fiecare stimuli extern si care reprezinta o estimare a încarcarii sistemului
- **ipoteza de eroare** - *fault hypothesis* - este un set de estimari referitoare la tipul de erori si la rata maxima de erori ce ar putea apare în timpul executiei, sistemul trebuind sa functioneze corect chiar în cazul aparitiei setului respectiv de erori.

Caracteristicile sistemelor TR

STR trebuie sa ia în considerare notiunea **timp** la toate nivelele.
În **proiectarea** unui **STR** trebuie sa fie adresate urmatoarele **aspecte** [Stankovic92]:

- Dimensiunea **timp** trebuie sa fie principiul central de proiectare. Constrângerile temporale trebuie sa fie surprinse de tehnicile de specificare si verificare folosite si deci sa nu fie tratate numai la implementare;
- Sa se realizeze echilibrul (fragil) dintre **flexibilitate s**i **predictibilitate**: sistemul trebuie sa ramâna suficient de flexibil pentru a se putea adapta unui mediu dinamic, dar în acelasi timp sa permita satisfacerea constrângerilor temporale;
- O gestionare a **resurselor** integrata care sa trateze aspectele legate de constrângeri temporale, predictibilitate, adaptabilitate, corectitudine, siguranta, toleranta.

Un **STR** trebuie sa ofere:

- un **model** care sa permita specificarea constrângerilor temporare pentru toate tipurile de procese
- un limbaj care sa permita specificarea clara a sistemului si care sa permita luarea în considerare a comunicatiilor asincrone cu exteriorul
- politici de **planificare s**i gestionare a resurselor care sa confere sistemului proprietatile de garantie si predictibilitate

- protocoale de **comunicatie** care sa ia în considerare aspectele temporare
- protocoale speciale pentru gestiunea **memoriei**
- mecanisme de **sincronizare** inter-taskuri si de sincronizare de ceas.

Aplicatiile TR din prima generatie rulau pe sisteme monoprocesor, problemele ce la rezolvau fiind relativ simple si nepresupunând algorimi sofisticati sau prelucrari foarte complexe. În ultimii 10-15 ani, sistemele TR au evoluat ca urmare a cercetarilor si rezultatelor deosebite din acest domeniu, care au fost imperativ cerute de necesitatea proiectarii unor aplicatii concrete, foarte complexe, critice, ce implica siguranta si predictibilitate deosebite: sisteme de control al traficului aerian, sisteme informationale distribuite, centrale nucleare, sisteme de aparare spatiale, largi sisteme de comanda si control în productie, etc.

Pentru a putea executa aplicatiile TR actuale, foarte complexe, **atributele sistemelor TR** trebuie sa fie:

- **Predictibilitatea** - se refera la garantarea îndeplinirii constrângerilor de timp ale aplicatiilor, în contextul dinamicii mediului;
- **Granularitati multiple ale taskurilor si de comunicatie** - aplicatiile TR complexe constau din taskuri de dimensiuni diferite, de la taskuri ce cuprind câteva instructiuni si care se executa cu o periodicitate ridicata, pâna la taskuri de dimensiuni foarte mari, care se executa rar. Taskurile comunica între ele prin mesaje de granularitati diferite;
- **Semantici diferite de timp** - Taskurile unui sistem pot avea grade diferite de criticalitate, urgenta, se pot modifica dinamic, pot exista grupuri de taskuri cu acelasi deadline, deci planificatorul trebuie trata toate aceste situatii;
- **Modele multiple de taskuri si comunicatii** - Constrângerile temporale ale taskurilor împlica constrângerea temporala a comunicatiilor; functie de modelul comunicatiilor utilizate, se poate presupune ca mesajele nu pot fi pierdute niciodata sau nesosirea unui mesaj la timp poate fi tolerata pentru a se putea trata un eveniment asincron.
- **Configurabilitate** - Din cauza arhitecturilor gazda care pot varia foarte mult de la sisteme cu un numar mic de procesoare la retele distribuite pe arii întinse - *WAN*, SOTR trebuie sa fie configurabile ca dimensiune si functionalitate;
- **Adaptabilitate** - SOTR trebuie sa se adapteze în performanta si functionalitate posibilelor schimbari externe. Cercetarile curente diferentiaza doua tipuri de adaptari: cele preventive, care anticipeaza schimbarile din mediu ai cele reactive, care trateaza schimbarile neasteptate.
- **Toleranta la defecte** - Sistemele trebuie sa încorporeze mecanisme performante software si hardware pentru detectarea si tratarea erorilor.

Performantele deosebite cerute de complexele sisteme TR actuale, precum si caracteristicile sistemelor distribuite, fac din sistemele distribuite o **solutie** viabila pentru sistemele TR.

Evolutia sistemelor TR

Prima propunere de utilizare a unui calculator pentru a opera în timp-real ca parte a unui sistem de control, a fost facuta de *Brown s*i *Campbell* în **1950** [BC50] de folosire de elemente analogice de calcul, neexcluzându-se si cele digitale.

Primele calculatoare digitale construite pentru control TR au fost cele pentru operarea avioanelor, iar în **1954**, un calculator *Digitrac* a fost cu succes folosit pentru zborul automat [Be94].

Aplicarea calculatoarea în controlul proceselor industriale s-a realizat la sfârsitul anilor '50: într-o electrocentrala din *Louisiana* si o rafinarie din Texas, prin *closed-loop* control.

Primul sistem de control digital direct a fost instalat în **1962** la o intreprindere chimica din *Anglia*; era un sistem cu 120 bucle de control si 256 puncte de masurare. În aceeasi perioada se înregistraza primul control ierarhic la o intreprindere din Texas.

Primele programe TR au fost scrise în cod masina, programele fiind relativ reduse. Cresterea în complexitate a aplicatiilor, a impus dezvoltarea unor sisteme de operare TR si a unor limbaje de programare TR. La sfârsitul anilor '60, au aparut primele compilatoare de *PROCESS FORTRAN*.

Minicalculatoarele aparute la începutul anilor '70 au fost folosite în sistemele TR, înregistrându-se si sisteme tolerante bazate pe solutia *stand-by*. Anul *1974*, de aparitie a **microprocesoarelor** a marcat o evolutie spectaculoasa în domeniul TR. S-a trecut de la sistemele TR monoprocesor, la cele multiprocesor si apoi la cele distribuite.

Proprietatea cea mai importanta a sistemelor in timp real: **comportarea predictibila**.
- Ideal: sistemul respecta toate teremele, chiar si la un maxim de incarcare.
- Majoritatea proiectantilor de SD gandesc in termenii
- Utilizatorilor independenti care acceseaza fisiere partajate aleator
- Agenti de calatorie care acceseaza o baza de date a liniilor aeriene care este partjata, la timpi imprevizibili.
- Este cunoscut ca daca este detectat un eveniment E, procesul X$$ trebuie sa ruleze, urmat de Y si Z, fie in ordine, fie in paralel.
- Mai mult, este adesea cunoscut (sau ar trebui sa fie cunoscut) care este comportarea cea mai proasta a acesto procese.
- De exemplu, este cunoscut ca X va necesita 50 msec, Y si Z necesita 60 msec fiecare, si pornirea procesului va dura 5 msec, atunci se poate garanta in avans ca sintemul va putea trata cinci evenimente periodice E per secunda in absenta oricarei alte sarcini.

§1.2. Elementele sistemelor în timp real

Se va considera ca exemplu o instalație pentru încalzirea aerului [Stu88] (figura 1.2-1).
Un ventilator sufla aer catre un element de încalzire, dirijarea facându-se printr-o conducta. La iesirea din aceasta este amplasat un termistor care formeaza un braț al unui circuit punte de masura. Iesirea amplificata a acestui circuit este proporționala cu temperatura si este o tensiune continua în domeniul 0..10 V care poate fi masurata în punctul B.
Valoarea curentului electric prin elementul de încalzire poate fi modificata printr-o unitate de control tiristorizata. Comanda acestei unitați se face cu o tensiune continua în domeniul 0..10V în punctul

A. Debitul aerului poate fi modificat prin intermediul unui obturator care este acționat de un motor reversibil. Motorul lucreaza la o viteza constanta iar pornirea sau oprirea sa poate fi comandata de un semnal numeric (0 sau 1) – aplicat la circuitul de control al motorului. Tot la acest circuit de control se aplica un semnal numeric pentru a comanda sensul de rotație. La obturator este atasat un potențiometru de pe care se poate culege o tensiune proporționala cu poziția obturatorului iar pozițiile „total închis” sau „total deschis” ale acestuia sunt detectate cu doua contacte. Operatorul are la dispoziție un panou de la care controlul instalației poate fi comutat pe Automat sau Manual. În modul de lucru Manual curentul de pe elementul de încalzire si poziția obturatorului (aspirație aer) pot fi modificate folosind doua potențiometre. Exista de asemenea comutatoare pentru a opera asupra ventilatorului si elementului de încalzire. Led-urile de pe panou indica ventilator pornit/oprit, element de încalzire oprit/oprit, obturator total deschis sau total închis si stare de funcționare (Automat sau Manual). În modul de lucru Automat prin controlul potențiometrelor poate fi ajustata temperatura elementului de încalzire si poziția obturatorului.

Controlul acestui proces simplu cu ajutorul calculatorului necesita **monitorizare** (urmarirea starilor si valorilor), **calcule** în conformitate cu algoritmul de control si **acționare**.

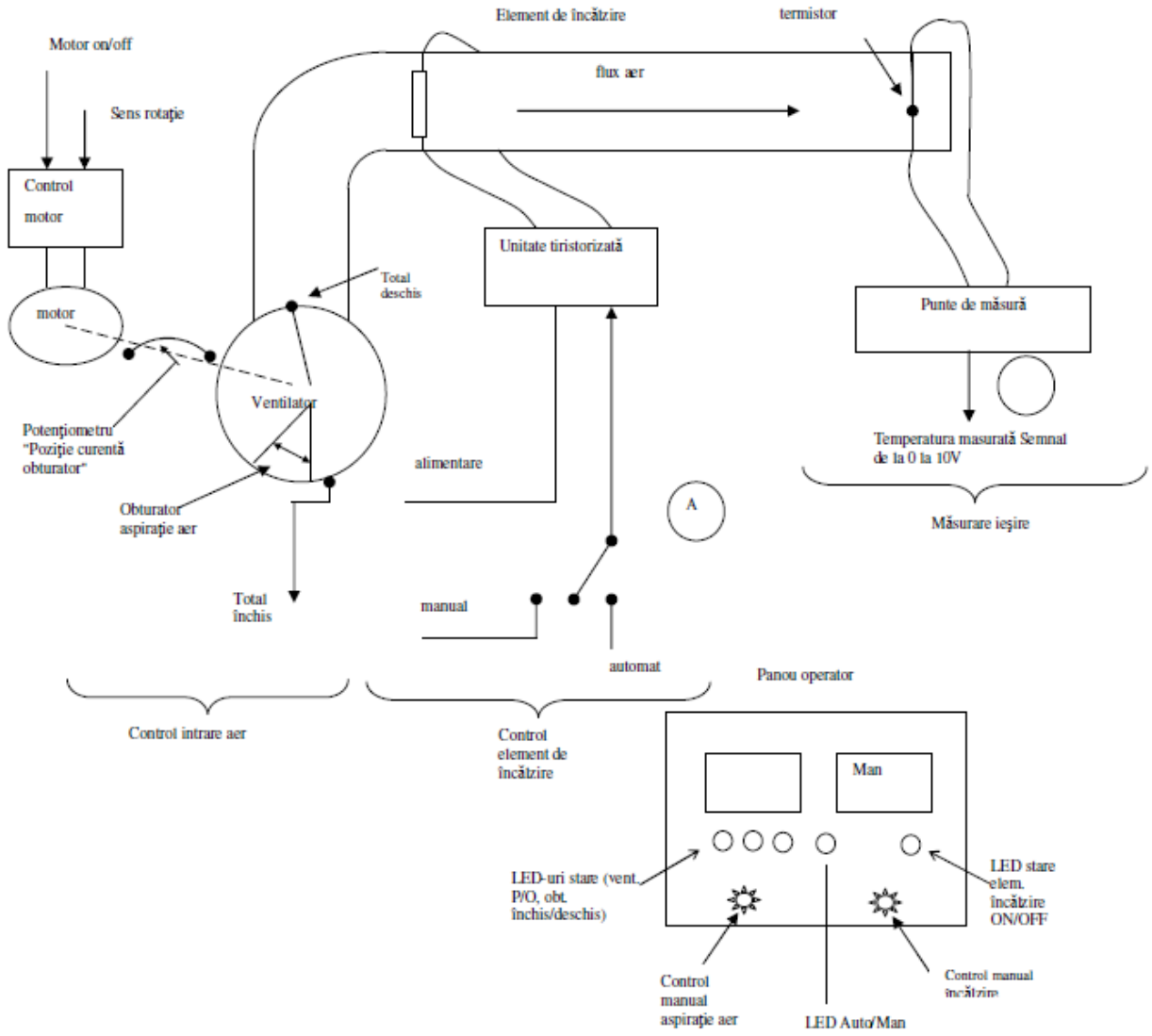


Fig. 1.2-1 Monitorizarea si controlul unei instalații de încalzire aer.

În figura 1.2-2 se prezinta o schema a sistemului de conducere cu calculator a instalației de încalzire aer.

Monitorizarea presupune obținerea informațiilor despre starea curenta a procesului. În exemplul prezentat aceasta se realizeaza prin intermediul semnalelor analogice precum temperatura aerului si poziția obturatorului, prin intrarile numerice de la pozițiile extreme ale obturatorului precum si prin semnale de stare – modul de lucru, motor pornit, încalzitor pornit etc..

Calculele sunt necesare pentru reglarea permanenta a temperaturii elementului de încalzire în corelație cu temperatura aerului la iesire. Reglarea se poate face în funcție de temperatura dorita prin intermediul obturatorului si prin elementul de încalzire. Calculele se refera si la rezolvarea unor situații conflictuale (interblocari):

- încalzirea rezistenței nu se poate face daca ventilatorul este oprit;
- comanda automata nu se poate face daca operatorul a trecut în mod de lucru manual iar comanda manuala nu se poate face daca modul de lucru este automat.

Acționarea presupune:

- furnizarea unei tensiuni pentru controlul elementului de încalzire în conformitate cu temperatura necesara la iesire;
- comanda iesirilor numerice pentru START/STOP motor, direcția de rotire;
- aprinderea ori stingerea ledurilor de la panou.

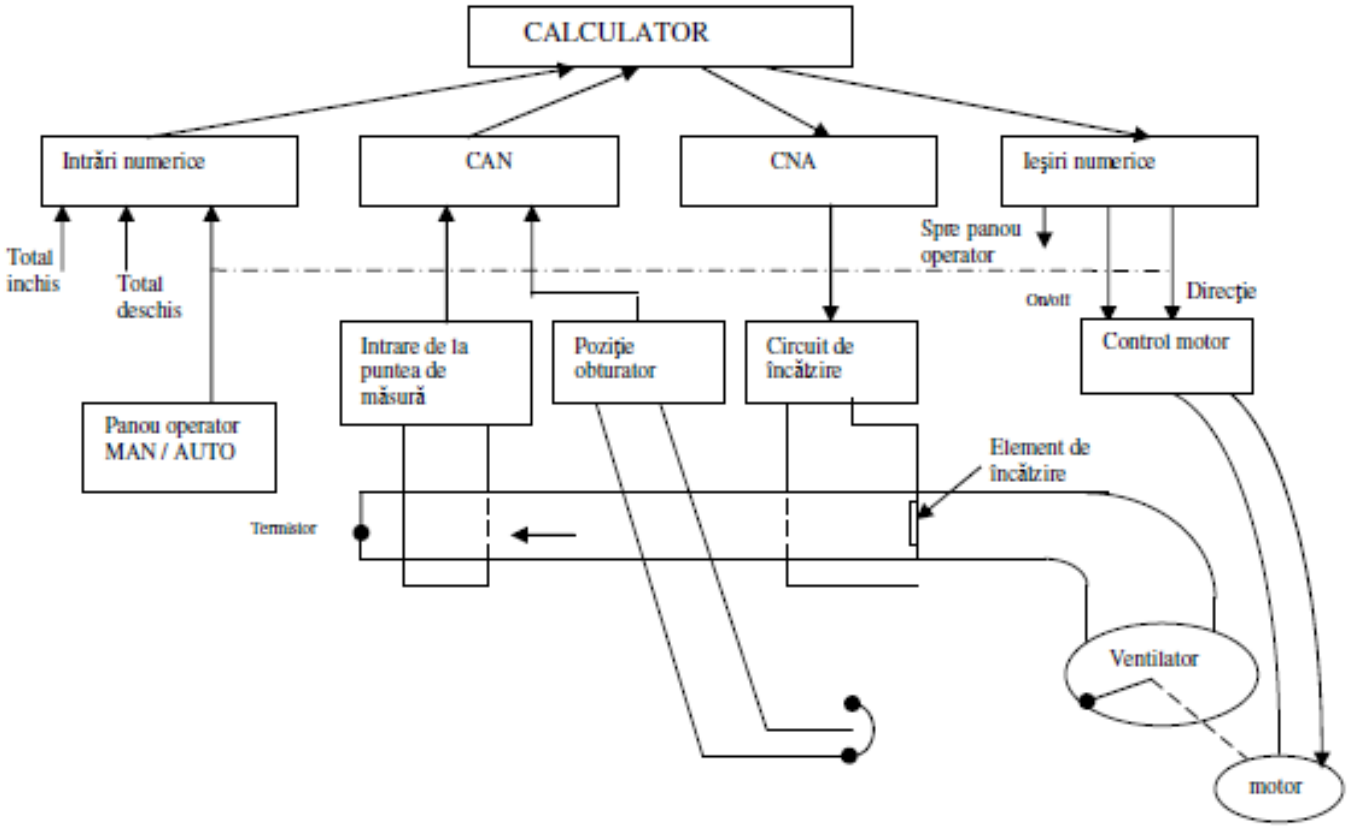


Fig. 1.2-2 Sistem de conducere cu calculator a instalaţiei de încalzire aer.

Programele (task-urile) de acţionare si monitorizare programeaza dispozitive de interfaţa precum convertor analog-numeric (CAN), convertor numeric-analogic (CAN), intrari numerice si iesiri numerice. Schema generala este prezentata în figura 1.2-3.

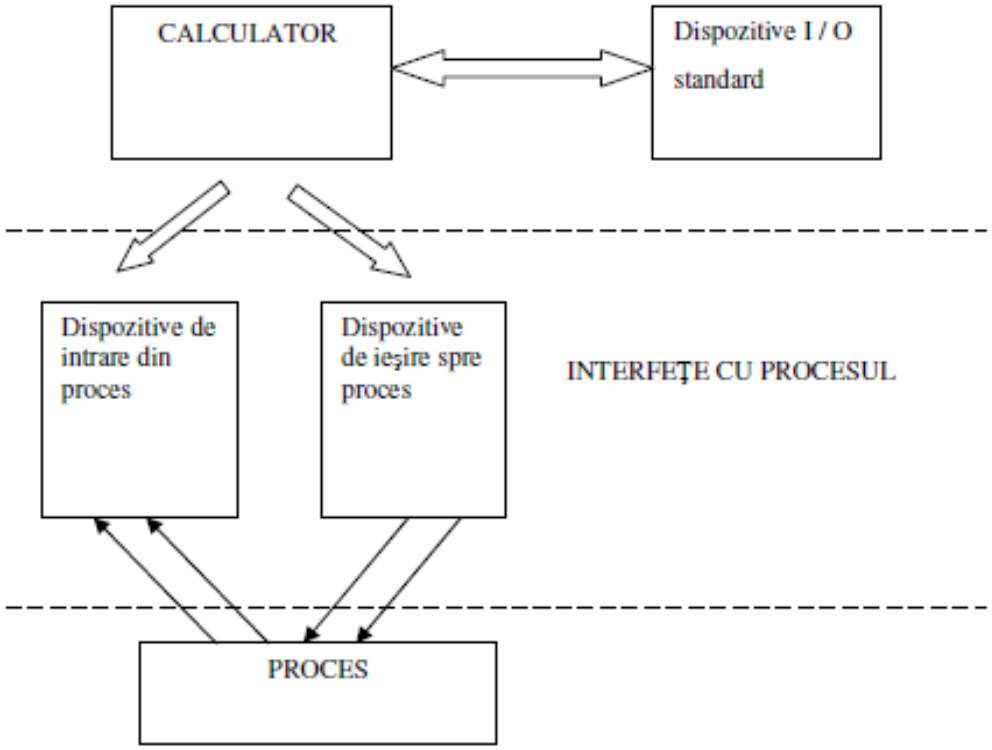


Fig. 1.2-3 Sistem generalizat de conducere cu procese.

Fiecare dintre tipurile de dispozitive necesita soft care sa opereze asupra lor. Ne vom referi la acest soft în continuare ca task-uri de intrare/iesire (I/E). În figura 1.2-4 se prezinta un sistem de control cu calculatorul care include interfețe hard si soft.

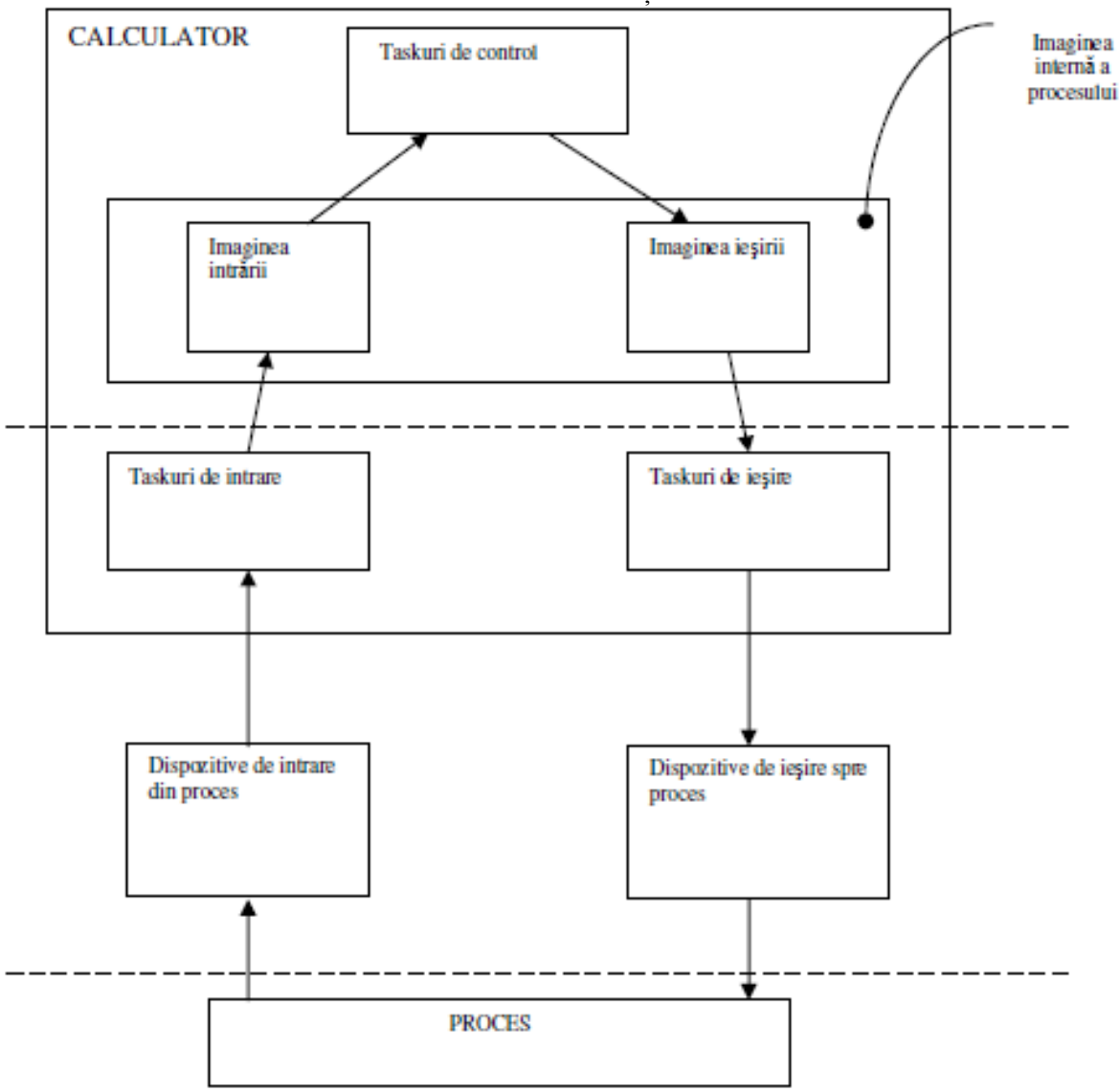


Fig. 1.2-4 Sistem de control generalizat care prezinta interfețe hard si soft

Dispozitivele de intrare si softul aferent furnizeaza informații pentru a crea o imagine interna a intrarilor dinproces. Imaginea intrarii este formata din esantioane prelevate din semnalele din proces astfel încât fenomenul de la intrare sa poate fi reconstituit fara distorsiuni. Ea este un instantaneu al starii procesului care trebuie actualizat:

• periodic (ciclic), la intervale de timp stabilite pre-execuție sau

• sporadic, la modificarea starii unor marimi discrete sau la modificarea valorii unor marimi continue în afara unui interval admisibil.

Imaginea iesirii reprezinta valori rezultate în urma aplicarii algoritmului de calcul. Aceasta este actualizata periodic de catre task-urile de control, intervalul de timp aferent perioadei fiind suficient de mic pentru a actualiza corect imaginea ieșirii. Task-urile de iesire transfera datele din imaginea ieșirii catre proces, sincronizat cu task-urile de control, acționând asupra acestuia astfel încât comportamentul sau sa corespunda imaginii ieșirii.

Acest model simplu de sistem de control descris împarte task-urile de realizat în trei grupe:

• task-uri de intrare in proces;

• task-uri de iesire spre proces;

• task-uri de control.

Comunicarea cu operatorul este tratata în schema ca parte a task-urilor de I/E. În multe aplicații comunicația reprezinta si altceva decât achiziție de date de la traductoare si senzori ori acționarea de comutatoare, valve etc. Sistemele de control pot fi distribuite pe mai multe calculatoare nu toate situate în acelasi loc si trebuie asigurata comunicația între acestea. Prin urmare, modelul prezentat anterior poate fi extins pentru a include si task-urile de comunicație (figura 1.2-5).

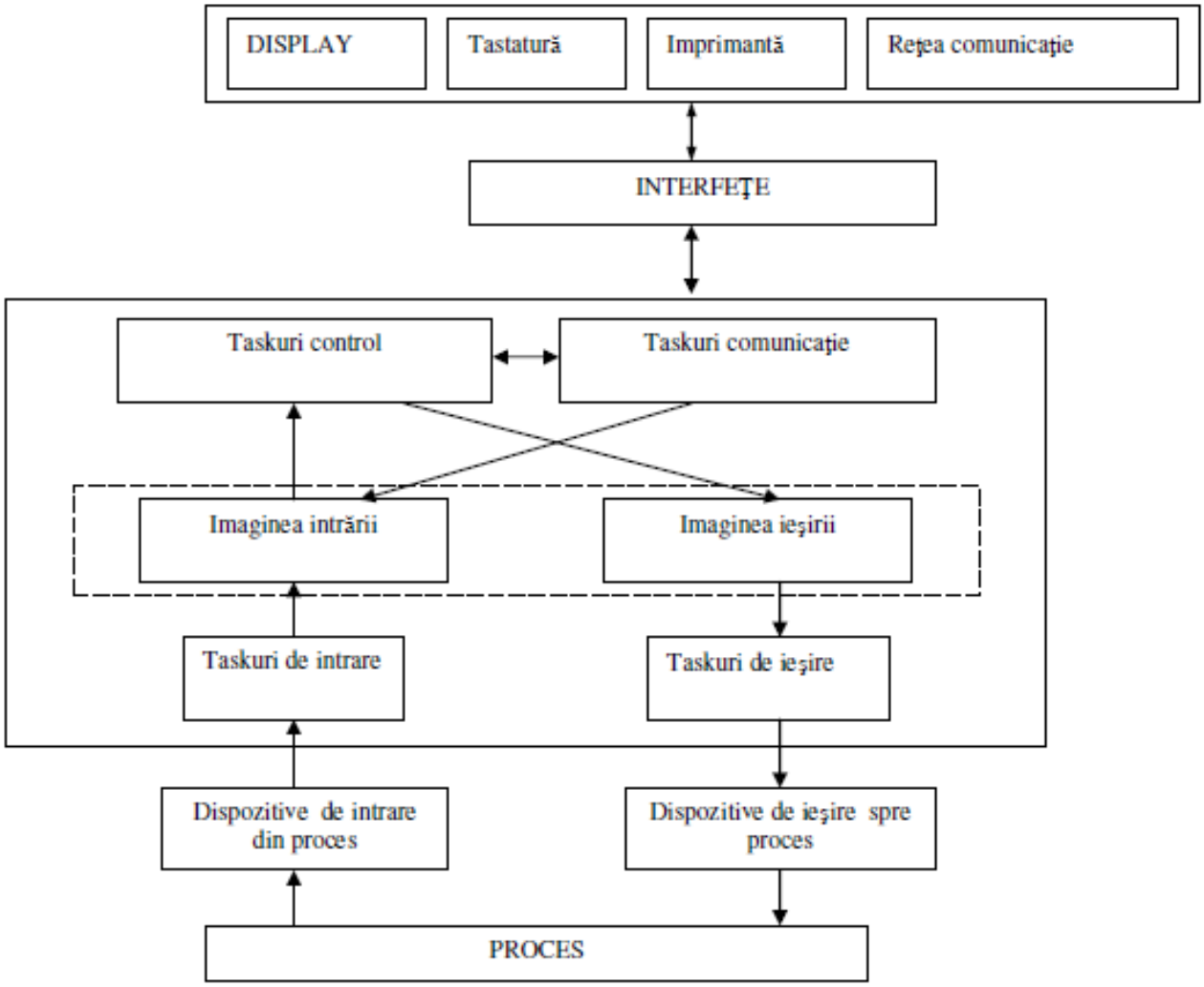


Fig. 1.2-5 Sistem de control generalizat, cu taskuri de comunicație.

§ 1.3 Tipuri de SCTR

Evolutia sistemelor timp real poate fi modelata utilizind o linie abstracta de timp care semnifica trecerea timpului din trecut inspre viitor. Orice aparitie care taie aceasta linie imaginara este un eveniment; de exemplu un ceas digital partitioneaza linia timpului intr-o secvente de timp de durate egale (perioada ceasului).

Un ***trigger*** este un eveniment care are ca rezultat executia unui task sau transmiterea unui mesaj. In functie de mecanismul adoptat pentru initierea comunicarii si activitatatii de procesare, exista doua abordari distincte care pot fi utilizate in procesul de proiectare a aplicatiilor timp real:

abordarea ***event-triggered*** **(ET*)*** – in cadrul acesteia toate activitatile din cadrul sistemului precum si comunicarea in sistem se realizeaza in urma aparitiei unor evenimente. Sesizarea evenimentelor semnificative se realizeaza utilizind binecunoscutul mecanism al intreruperilor si in consecinta este necesara aplicarea unei strategii de planificare dinamica pentru activarea task-ului corespunzator tratarii evenimentului aparut

abordarea ***timed-triggered*** **(TT)** – in cadrul careia toate activitatile, inclusiv comunicarea, sunt initiate la momente predefinite de timp, existind o singura intrerupere, si anume intreruperea de ceas. Daca sistemul este distribuit atunci se presupune ca ceasurile fiecarui nod sunt sincronizate intre ele conform unui ceas global si ca fiecare eveniment poate fi **stampilat(timestamped)** in functie de acest timp. Granularitatea ceasului global trebuie astfel aleasa incit ordinea in timp a aparitiei evenimentelor intr-un sistem distribuit sa poata fi stabilita pe baza **sampilei de timp** a acestora.

Astfel, daca sincronizarea dintre procesele externe si task-urile interne este definita in functie de timpul curent, inseamna ca la proiectarea sistemului in cauza a fost utilizata abordarea **TT**; daca insa aceasta se defineste in functie de anumite evenimente, s-a utilizat abordarea **ET**. In practica exista si o a treia categorie de sisteme, asa numite **interactive**, in cadrul carora relatiile intre actiunile din cadrul computerului si cele externe sunt mai putin strict definite.

Avantajele si dezavantajele abordarilor **TT** versus **ET** pentru sisteme timp real hard trebuiesc bine cunoscute in cazul modelarii unui astfel de sistem; functie caracteristicile intrinseci ale aplicatiei propriu-zise, se alege abordarea care se preteaza cel mai bine in contextul performanta/cost. Indiferent insa de calea aleasa, un model al sistemelor in timp real trebuie sa se bazeze pe un set general valabil si bine definit de concepte.

Astfel, intr-un sistem ***event-triggered*** **ET** se utilizeaza mecanismul intreruperilor pentru tratarea evenimentelor externe. Cind un astfel de mecanism este activat si apare o intrerupere, executia task-ului curent este intrerupta si hardware-ul forteaza o schimbare de context catre rutina de tratare a intreruperii corespunzatoare. Dupa terminarea tratarii intreruperii, o noua schimbare de context trebuie realizata fie pentru a continua task-ul intrerupt, fie pentru a trata o alta intrerupere aparuta intre timp.

Aceste schimbari de context consuma un anumit timp, asa numitul **WCAO (Worst Case Administrative Overhead)**. Aparitia fiecarei intreruperi reduce capacitatea unitatii centrale disponibila aplicatiei, dupa cum este ilustrat in Figura 1.3. Acest lucru este valabil si in cazul in care rutina de tratare decide ca intreruperea a fost eronata si nu activeaza nici un alt task.

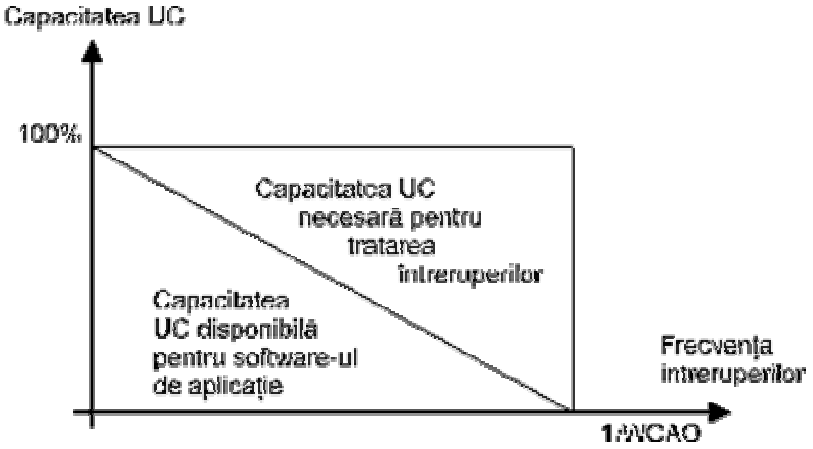


Figura 1.3. Capacitatea CPU versus 1/WCAO

Se observa din Figura 1.3 ca daca frecventa intreruperilor atinge valoarea 1/WCAO, capacitatea unitatii centrale disponibila pentru aplicatii devine zero; este deci de o importanta covirsitoare ca frecventa intreruperilor intr-un sistem timp real sa fie limitata. Acest lucru este practic imposibil intr-un sistem ET, deoarece sursa intreruperilor se afla in afara sferei de control a calculatorului.

In cazul unui sistem timp real ***timed-triggered*** **TT** insa, controlul ramine intotdeauna in cadrul calculatorului. Pentru a sesiza un anumit eveniment exterior, un sistem **TT** trebuie sa utilizeze

un task periodic ***time-triggered*** care sa evalueze starea variabilelor din sistem. Perioada acestui task trebuie sa fie mai mica decit **laxity**-ul (diferenta intre deadline si timpul de executie) a oricarei tranzactii in timp real care poate fi activata de un anumit eveniment. Si utilizarea unui task trigger genereaza timpi suplimentari in sistem; practic, daca **laxity**-ul unei tranzactii este foarte mic (<1ms), overhead-ul asociat task-ului trigger poate fi mult prea mare pentru a fi admisibil.

In concluzie, trebuie in mod obligatoriu precizat faptul ca, pentru sistemele care abordeaza modalitatea ***event-triggered*** **ET** si lucreaza prin intreruperi, predictibilitatea este un deziderat greu de atins. De asemenea, o alta cerinta, si mai greu de atins dar esentiala pentru sistemele critice, aceea de determinism, nu poate fi practic realizata. De altfel, orice modalitate de a schimba in mod dinamic (de exemplu, conform prioritatii) ordinea proceselor in astfel de sisteme rezulta intr-un sistem nedeterministic. Rezultatele obtinute de Liu si Layland (cap. 6.1.2.1) privind algoritmul de planificare rate-monotonic pot fi utilizate doar pentru dovedirea predictibilitatii; ele insa nu rezolva problema determinismului.

Exista la ora actuala numeroase sisteme timp real care utilizeaza atit abordarea **ET** cit si abordarea **TT**. Totusi, majoritatea arhitecturilor tind sa favorizeze mai mult una dintre ele: semnalele de control sunt predominant event-triggered sau predominant time-triggered. O comparatie intre cele doua variante poate fi sumarizata conform Tabelului 1.

Tabelul 1: Activarea task-urilor prin intreruperi versus utilizarea unui task trigger

Caracteristica	**Intrerupere**	**Task trigger**
Factori care afecteaza latenta	raspunsul la intrerupere	perioada de achizitie
Sursa controlului	externa	interna
Overhead-ul datorat scanarii	nu exista	WCET(laxity_trigger)/ laxity
Overhead-ul datorat procesarii	variabil	constant
Posibilitatea preemtiunii	oricind	controlata
Conditiile de trigger	simple	complexe
Predictabilitate	nivel scazut	nivel ridicat

§ 1.4 Clasificarea programelor

Studii experimentale arata clar ca anumite tipuri de programe, în particular acelea care realizeaza operații timp - real, sunt mult mai dificil de realizat decât programele obisnuite. Tehnicile de lucru pentru verificarea corectitudinii programelor sunt date si de diferențele între diferitele tipuri de programe.

Lucrari teoretice referitoare la tehnicile pentru demonstrarea corectitudinii unui program au permis identificarea a 3 tipuri de programe:

- secvențiale;
- multitasking;
- timp - real.

1.4.1 Programele secvențiale

Acțiunile sunt strict ordonate ca o secvența în timp. Comportarea unui program depinde numai de efectele instrucțiunilor individuale si de ordinea lor. Timpul necesar pentru o anumita prelucrare este mereu acelasi.

Pentru testare se porneste de la urmatoarele premise:

1. O instrucțiune practic defineste o acțiune fixa (statica) în sistem;
2. Oricare dintre acțiunile programului produc secvențe statice de evenimente.

1.4.2 Programe multitasking

Difera de cele secvențiale prin aceea ca operațiile care trebuie realizate *operațiile care trebuie realizate nu sunt executate neaparat ca o secnvența continua în timp.*. Mai mult, operațiile se pot desfasura concurent. Astfel, un program poate fi construit dintr-un numar de parți numite „procese" sau „task"-uri care:
• luate separat, sunt compuse din secvențe de instrucțiuni;
• se executa concurent cu alte task-uri;
• comunica cu alte task-uri prin memorie partajata, semnale de sincronizare, mesaje, cutii postale... Testarea presupune aceleasi premise ca la programele secvențiale, cu unele deosebiri:
• Task-urile pot fi testate separat numai daca variabilele fiecarui task sunt distincte. Daca variabilele sunt partajate, posibila concurența la resurse poate suspenda execuția pâna la eliberarea acestora;
• Cerințele de sincronizare cu alte task-uri fac ca timpul de execuție a secvenței de instrucțiuni pentru fiecare task sa nu poata fi determinat cu precizie de catre procedurile de validare - timpul de execuție a task-urilor depinde si de modul de execuție a procedurilor de sincronizareț
• Prioritatea de lansare în execuție: timpul de executie al task-ului este influențat ți de intervalul de timp în care execuția sa este suspendata atunci când procesorul executa alte task-uri cu prioritate mai mare.

1.4.3 Programe timp - real

Un program timp - real difera de tipurile anterioare de programe prin aceea ca pe lânga faptul ca acțiunile sale nu sunt disjuncte în timp, secvența acțiunilor sale nu este determinata doar de proiectant, ci ți de evenimentele din mediul exterior. Acestea sunt declansate de condiții din afara calculatorului si nu pot fi facute sa respecte, de exemplu, regulile de sincronizare dintre task-uri!

Un program timp - real poate fi împarțit în task-uri, dar comutarea dintre acestea nu asteapta neaparat dupa un semnal intern de sincronizare, deoarece ceea ce se întâmpla în mediul extern nu poate fi întârziat. În programul timp - real, în contrast cu celelalte tipuri de programe, timpul actual asociat cu o acțiune este un factor esențial în procesul de validare. Mijloacele de determinare a corectitudinii execuției difera fundamental de mijloacele asociate programelor secvențiale sau multitasking.

Multe limbaje de programare folosite uzual nu poseda mecanisme pentru realizarea aplicațiilor timp - real, iar programatorul trebuie sa si le creeze singur. De asemenea, instrumentele de testare utilizate în mod curent pentru validarea programelor nu funcționeaza corect în aplicații timp - real. Chiar daca exista limbaje de programare pentru aplicații timp - real, acestea sunt elaborate pentru anumite tipuri de masini/sisteme de operare, portarea pe alte platforme hard/soft necesitând eforturi nu tocmai neglijabile. Aceasta determina un cost al aplicației mult mai ridicat, iar în multe situații lipsa unor mijloace de dezvoltare ți testare accesibile este în contrast evident cu necesitatea unor aplicații fiabile.

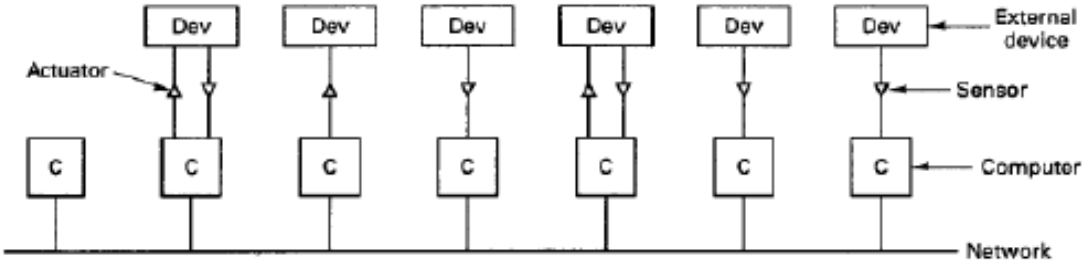
Au fost realizate sisteme de operare de TR, dar datorita efortului mare necesar pentru realizarea lor si a pieței relativ limitate, acestea au fost completate si cu module specifice sistemelor de operare de uz general; exemple de SOTR timp - real : QNX, LYNX, VxWorks, RT Linux, eCOS.

Uneori, se obisnuieste ca pentru aplicații timp - real sa fie realizate de proiectanții de aplicații *executive de timp real dedicate*, care implementeaza algoritmi de planificare si sincronizare proiectați în funcție de aplicație.

§ 1.5 Definirea si caracteristicile sistemelor distribuite

Un **sistem distribuit** este definit ca fiind un sistem constituit din mai multe elemente de prelucrare - **noduri** - legate în retea, care coopereaza pentru realizarea unei prelucrari integrate.

Sistemele distribuite in timp real pot fi adesea structurate ca in fig de mai jos.
 Vedem o coletie de calculatoare conectate in retea.
 Unele dintre acestea sunt conectate la dispozitive externe care produc sau accepta date si se asteapta a fi controlate in timp real.
 Calculatoarele pot fi microcontrolere mici incastrate in dispozitive sau masini de sine statatoare
 In ambele cazuri in mod uzual au senzori pentru receptionarea de semnale de la dispozitive digitale sau analogice pentru expedierea semnalelor catre ele.



Caracteristicile sistemelor distribuite sunt [CDK95]:

- partajarea resurselor
- deschidere - sistemul poate fi extins prin noi servicii fara o întrerupere (disruption) sau duplicare a serviciilor existente)
- concurenta
- scalabilitate
- toleranta la defecte
- transparenta - transparenta distribuirii cuprinde entitatile: accesul la informatii, localizarea, concurenta, replicarea, defectele, migrarea, performanta si scalarea.

Sistemele distribuite pot fi:

- **puternic conectate** - *tightly coupled* - daca nodurile componente au acces la o memorie comuna
- **slab conectate** - *loosely coupled* - daca nodurile componente nu au acces la o memorie comuna.

De asemenea, functie de **tipurile nodurilor**, sistemele distribuite pot fi:

- **omogene** - daca toate procesoarele componente sunt de acelasi tip
- **neomogene** - daca procesoarele componente nu sunt de acelasi tip.

Avantajele distribuirii pentru domeniul TR sunt:

- performanta îmbunatatita prin exploatarea paralelismului
- cresterea disponibilitatii (*availability*) si sigurantei (*reliability*) datorita redundantei
- dispersia prelucrarii în punctele de comanda si control
- facilitatea de crestere incrementala prin adaugarea de procesoare si linii de comunicatie.

De cele mai multe ori, aplicatiile TR au ca suport sisteme distribuite slab cuplate, omogene.

Distribuirea aplicatiilor TR, aduce însa o serie de **probleme noi** proiectarii sistemelor TR, fiecare concretizându-se în câte un domeniu actual de cercetare:

- tehnici de specificare si verificare a constrângerilor temporale si de distribuire
- planificarea distribuita a aplicatiei
- sincronizare globale de ceas, noi mecanisme de sincronizare si comunicare între taskuri
- limbaje de programare distribuita în TR
- toleranta la defecte în mediu TR si distribuit.