Cerinţe impuse echipamentelor numerice pentru conducerea proceselor

Pentru a fi utilizate în domeniul conducerii proceselor, calculatoarele numerice trebuie să răspundă următoarelor cerinţe:

- să prezinte o siguranţă ridicată în funcţionare;
- să asigure procesarea în timp real a datelor aferente procesului condus;
- să permită conectarea la traductoarele şi elementele de execuţie din proces;
- să poată fi operate de către personalul de operare al procesului.

1. Siguranţa în funcţionare

Desfăşurarea în condiţii de siguranţă a unui proces este nemijlocit influenţată de siguranţa în funcţionare (*SF*) a echipamentului de conducere (*EC*).

În conformitate cu Standardul ISO 8402/86 *calitatea reprezintă totalitatea proprietăţilor şi caracteristicilor unui produs sau serviciu care îi conferă acestuia aptitudinea de a satisface anumite cerinţe exprimate sau implicite.* Pornind de la această definiţie rezultă clar că *SF* este o componentă a calităţii. *SF a EC* este dictată de *fiabilitate, mentenabilitate şi disponibilitate* prin intermediul indicatorilor asociaţi.

Fiabilitatea se defineşte din punct de vedere *calitativ* ca fiind aptitudinea unui produs de a îndeplini corect funcţiile prevăzute un anumit timp în condiţii de exploatare specificate.

Referitor la această definiţie a fiabilităţii se pot face următoarele observaţii:

- produsul trebuie sa conserve performanţele constatate sau specificate în procesul de fabricaţie;

- performanţele specificate se păstrează pe o durată cel puţin egală cu durata de serviciu a produsului;

- funcţiile prevăzute sunt realizate dacă exploatarea se face în condiţii specificate.

Cel mai important indicator pentru caracterizarea fiabilităţii este *Timpul între defecţiuni* **(*MTBF - Mean Time Between Failures*)**. Acesta reprezintă media duratelor de bună funcţionare pentru numărul de produse luate în considerare.

Pe baza determinărilor experimentale *MTBF* se calculează cu relaţia:

$$MTBF = \frac{\sum_{i=1}^{N0} t_{fi}}{N_0}$$

unde: t_{fi} este timpul mediu de funcţionare a unui produs din lot;
N_0 - mărimea lotului.

Alături de fiabilitate *SF* a unui produs mai este caracterizată şi de

Mentenabilitatea (*M*). se defineşte din punct de vedere calitativ ca fiind aptitudinea unui produs de a fi menţinut sau repus în funcţiune în condiţii prescrise.

M este asigurată prin intermediul suportului de mentenanţă (activitate de service) care necesită următoarele specificaţii:

- timpii de intevenţii (reparaţii şi revizii) şi de staţionare pentru efectuarea acţiunilor de mentenanţă corectivă sau preventivă;

- numărul şi nivelul de calificare al personalului implicat în acţiunea de service;

- numărul de subansamble, piese de schimb şi SDV-uri necesar în stocul de rezervă;

- măsuri de protecţie a muncii pe parcursul desfăşurării operaţiilor de mentenanţă.

Principalul indicator al mentenabilităţii este *Media timpilor de reparaţie* **(*MTR* - *Mean Time Reparation*).**

Disponibilitatea unui echipament (*D*), reprezintă dinin punct de vedere *calitativ* aptitudinea acestuia de realiza funcţiile specificate sub aspectul combinat al fiabilităţii şi mentenabilităţii la un moment dat sau un timp specificat.

Uzual disponibilitatea unui produs se apreciază prin intermediul indicatorilor, între care cel mai este coeficientul de disponibilitate, care se calculează cu relaţia de mai jos

$$K_D = \frac{MTBF}{MTBF + MTR} \cdot 100,$$

Practic K_D reprezintă proporţia din durata de serviciu prevăzută a unui echipament, în care acesta este disponibil sub aspectul combinat al *fiabilităţii* şi *mentenabilităţii*

2. Procesarea datelor în timp real

Funcţiile principale ale unui sistem de conducere se referă la:

- preluarea informaţiei tehnologice;
- prelucrarea acesteia în vederea elaborării comenzilor;
- transmiterea comenzilor către proces.

Un sistem de conducere are comportare în timp real dacă viteza de reacţie la schimbările din proces este în concordanţă cu inerţia acestuia. Comportarea în timp real presupune un sincronism care trebuie să existe între operaţiile interne de calcul şi evenimentele lumii exterioare.

Pornind de la diversitatea proceselor (din punct de vedere al inerţiei) rezultă valori diferite ale bazei de timp asociate timpului real. Astfel dacă pentru procesele de transfer masic sau energetic sunt suficiente baze de timp de ordinul secundelor, în acţionările electrice sau în comanda roboţilor aceste baze de timp trebuie să fie de ordinul zecimilor de secundă sau chiar mai mici.

Comportarea în timp real la *achiziţia datelor* implică obţinerea informaţiei aferente într-un interval de timp inferior celei mai mici constante de timp a procesului. Informaţia privind parametrii procesului poate fi destinată prelucrării într-un algoritm de conducere şi/sau vizualizării pe ecranele consolei operatorului de proces. În fiecare din situaţii informaţia trebuie să devină disponibilă înainte ca datele din proces să-şi piardă consistenţa.

Transmiterea în timp real a comenzii către proces presupune implementarea acesteia înainte ca informaţia pe baza căreia a fost determinată să-şi piardă valabilitatea.

Facilităţile de lucru în timp real sunt asigurate de componente ale infrastructurii hardware şi software ale sistemului de conducere.

Resursa hardware principală este reprezentată de *ceasul de timp real* cu ajutorul căruia se poate realiza fie măsurarea unor intervale de timp fie execuţia unor funcţii la momente de timp precizate. Ceasul de timp real poate opera fie prin *interogare* fie în *conjuncţie* cu sistemul de întreruperi. Operarea prin interogare presupune citirea timpului curent prin program iar lucrul în întreruperi presupune generarea unui semnal de întrerupere la expirarea unui interval de timp specificat.

Din punct de vedere software disponibilităţile de lucru în timp real sunt asigurate de *sisteme de operare* adecvate iar pentru dezvoltarea de aplicaţii, de *limbaje specializate.*

În concluzie cerinţa de prelucrare în timp real se referă la desfăşurarea operaţiilor de calcul în sincronism cu evenimentele lumii exterioare. Cuanta de timp asociată timpului real nu este universală ci este legată de inerţia procesului. Ca elemente de infrastructură care fac posibilă prelucrarea în timp real sunt de menţionat *ceasul de timp real, sistemul de întreruperi, sistemul de operare*.

3. Conectarea la perifericele de proces

Utilizarea echipamentelor numerice de calcul în conducerea proceselor presupune un permanent schimb de informaţie între cele două entităţi.

Informaţia privind starea procesului se obţine prin intermediul *traductoarelor* iar transpunerea în proces a comenzilor generate de către echipamentul numeric este realizată de către *elementele de execuţie.*

Pentru echipamentul numeric de conducere procesul reprezintă unul din utilizatori. Pentru acest utilizator *special* traductoarele facilitează introducerea informaţiei în echipamentul numeric, iar elementele de execuţie permit preluarea informaţiei de la acesta. Având în vedere consideraţiile de mai sus, traductoarele şi elementele de execuţie reprezintă echipamente periferice de un tip deosebit, care în continuare vor fi numite *echipamente periferice de proces (EPP).*

Multitudinea problemelor ce se cer a fi rezolvate de către sistemele automate aflate în legătură nemijlocită cu procesul implică existenţa unei mari diversităţi de *EPP.* Indiferent cărui tip aparţin *EPP,* există o totală incompatibilitate între semnalele specifice acestora şi cele cu care operează în mod curent echipamentele de conducere.

Pentru a putea fi conectat la *EPP* un echipament de conducere trebuie să conţină o *interfaţă* adecvată. Un sistem de interfaţă cu procesul *(SIP)* conţine două subsisteme destinate funcţiilor de *achiziţie* a semnalelor furnizate de traductoare şi respectiv de *generare* a semnalelor de comandă către elementele de execuţie.

Între cerinţele importante impuse unui SIP sunt de menţionat următoarele:

- transmiterea semnalelor să se realizeze fără alterarea conţinutului informaţional al acestora;

- rejectarea zgomotelor datorate mediului industrial;

- protejarea echipamentului numeric prin realizarea în măsura posibilităţilor a separării galvanice.

4. Dialogul cu personalul de operare

Conducerea instalaţiilor tehnologice se realizează din camere sau puncte de comandă. În aceste locuri trebuie să existe mijloace capabile să ofere operatorului de proces posibilităţi care să-i permită atât informarea privind starea procesului cât şi intervenţii ocazionate de anumite evenimente apărute în evoluţia acestuia.

În cadrul echipamentelor de conducere aceste facilităţi sunt oferite de *consola operatorului de proces (COP).*

Uzual în structura unei *COP* intră ecrane, tastaturi şi mai rar chei, butoane, lămpi etc. În ceea ce priveşte ecranele acestea trebuie să fie cu o diagonală suficient de mare (minimum 21 inches) pentru ca informaţia oferită să poată fi lejer interpretată de către operator. Având în vedere că informaţiile privind starea procesului sunt oferite preponderent sub formă grafică rezoluţia ecranului trebuie să fie corespunzătoare.

În ceea ce priveşte tastaturile acestea trebuie să conţină un număr relativ restrâns de taste, de regulă funcţionale. În mod obişnuit fiecărui ecran îi este asociată o tastatură situată, de regulă, în vecinătatea nemijlocită a acestuia.

Din punct de vedere funcţional un ansamblu ecran-tastatură poate grupa mai multe funcţii pentru o anumită secţiune din instalaţie sau poate realiza o singură funcţie pentru întreaga instalaţie. Când se vorbeşte, în această situaţie, de funcţii se au în vedere încadrări de tipul: *supraveghere şi alarmare, stări bucle de reglare, evoluţie în timp parametri, comenzi utilaje dinamice* etc.

Sistemul de programe asociat *COP* va fi astfel realizat încât să permită operarea acesteia de către un personal ce posedă o calificare minimă în domeniile informatic şi al tehnicii de calcul.

COP în calitate de componentă a echipamentului de conducere trebuie să satisfacă şi cerinţele legate de siguranţa în funcţionare şi de prelucrarea în timp real.

În concluzie *COP* creează facilităţi de comunicare cu EC pentru personalului de operare care, de regulă, cunoaşte foarte bine procesul dar deţine cunoştinţe minime asupra echipamentului.

Operaţii fundamentale multitasking

Concurenţa pentru deţinerea resurselor poate genera situaţii conflictuale între taskuri. Pentru evitarea acestora şi pentru ca taskurile să-şi realizeze obiectivele au fost formalizate aşa numitele *operaţii multitasking* între care o importanţă aparte prezintă: *excluderea mutuală, sincronizarea şi comunicarea* [1]

Pentru implementarea acestor operaţii sistemele de operare sau executivele de timp real pun la dispoziţie instrumente cum ar fi: *semafoare, cutii poştale, mesaje de trecere, variabile de tip eveniment, fanioane de excluziune, monitoare, etc.*

1. Resurse şi secţiuni critice

În general termenul de *resursă* desemnează orice element necesar unui task pentru a putea fi executat . Resursele pot fi de două categorii şi anume *materiale* şi *logice.*

În rândul resurselor materiale pot fi considerate:

- procesorul;
- memoria internă;
- memoriile externe;
- dispozitivele de intrare / ieşire;
- ceasul de timp real;
- dispozitivele speciale.

În resursele logice pot fi incluse:

- subrutinele (procedurile);
- masivele de date;
- fişierele;
- variabilele.

O altă clasificare împarte resursele în: *resurse locale* şi *resurse comune.*

- Resursele locale sunt resursele care pot fi *accesate de către un singur task.* Acestea pot fi în primul rând de natură logică (*subrutine, fişiere, variabile*), dar pot fi şi de natură fizică, cum ar fi anumite dispozitive de intrare / ieşire.

[1] În mod obişnuit acestea sunt cunoscute ca *Operaţii Fundamentale Multitasking.*

- Resursele comune sunt acele resurse care pot fi *accesate de mai multe taskuri.* Acestea pot fi atât de natură *logică (subrutine, tampoane de date, variabile, etc.)* cât şi fizică *(procesor, memorie, dispozitive de intrare/ieşire etc.).*

La rândul lor resursele comune pot fi *critice, partajabile, reentrante.*

- O resursă comună asupra căreia la un moment dat se poate desfăşura o singură *sesiune de lucru* respectiv care poate fi accesată la un moment dat de către un singur task se numeşte ***resursă critică.*** În această categorie pot fi incluse *procesorul, locaţiile de memorie, echipamente de intrare, subrutinele care îşi modifică pe parcurs unele date proprii, etc.*
- O resursă comună care admite ca ***n*** *sesiuni de lucru* să fie în derulare asupraei la un moment dat *(respectiv să poată fi accesată de către* ***n*** *taskuri)* se numeşte ***resursă partajabilă*** *cu* ***n*** *puncte de intrare.*
- O resursă comună care admite să fie *oricâte sesiuni de lucru* să fie în derulare la un moment dat se numeşte ***resursă reentrantă***[2]**.**

În continuare vor fi tratate unele aspecte referitoare la utilizarea *neconflictuală* a resurselor critice. Secţiunea dintr-un task în care este accesată o resursă critică se numeşte ***secţiune critică.*** În sistemele multitasking trebuie să existe reglementări pentru accesarea *secţiunilor critice.* În acest sens se impune existenţa unor modalităţi de implementare a *excluderii mutuale.*

Excluderea de către un task aflat într-o secţiune critică referitoare la o resursă a accesului altor taskuri de a accesa propriile *secţiuni critice referitoare la aceiaşi resursă* se constituie în ***excludere mutuală.***

Referitor la implementarea *excluderii mutuale* sunt de menţionat recomandările făcute de prof. Andrew Tanenbaum de la Universitatea Vrijie Amsterdam, Olanda în anul 1987.

Sintetic aceste recomandări au în vedere următoarele aspecte:

1 – orice secţiune critică poate fi executată la un moment dat de către un singur task (aceasta este practic esenţa *excluderii mutuale* care afirmă că un singur task se poate afla la un moment dat în propria secţiune critică referitoare la o resursă);

2 – nu se poate face nici o ipoteză în ceea ce priveşte viteza relativă şi frecvenţa de execuţie a taskurilor (cu alte cuvinte la implementarea *excluderii mutuale* nu pot fi avute în vedere considerente legate de frecvenţa sau viteza de execuţie a taskurilor);

[2] Resursele *reentrante* pot fi considerate resurse *partajabile* cu *n* tinzând către infinit.

3 – orice task are dreptul să acceseze propria *secţiune critică* referitoare ala o anumită resursă după un interval finit de timp;

4 – orice task trebuie să evacueze o *secţiune critică* proprie după un interval finit de timp;

5 – taskurile nu se pot bloca în interiorul propriilor secţiuni critice.

În continuare vor fi prezentate câteva modalităţi de implementare a *excluderii mutuale.*

2. Excluderea mutuală realizată cu semafoare

Un *semafor* reprezintă, conform introdus în anul 1965 de către matematicianul olandez Edsger Wybe Dijkstra, un dublet format dintr-o variabilă de tip întreg *I* şi o coadă de aşteptare *C*, respectiv

$S = (I, C)$.

La iniţializare variabilei I i se atribuie o valoare $I_0 \geq 0$, iar coada C este vidă. Dacă variabila *I* ia numai valorile *0 şi 1* semaforul se numeşte *binar* iar dacă ia şi alte valori întregi, semaforul se numeşte *general.*

Asupra semafoarelor pot fi efectuate două operaţii denumite ***P*** (de la cuvântul olandez ***P****asseren – a trece)* şi ***V*** (de la cuvântul olandez ***V****rigeven – a elibera).* Aceste funcţii sunt indivizibile[3] şi se numesc *primitive.* În ceea ce priveşte *coada de aşteptare* , aceasta poate gestionată conform principiului *FIFO*[4], sau după priorităţi (univoce sau neunuivoce).

Primitiva *P(S)* presupune realizarea următoarelor operaţii (ilustrate în organigrama din figura 1):

1 - *I←I-1;*

2 – dacă $I \geq 0$, funcţia se încheie şi taskul în care aceasta se execută îşi continuă execuţia;

3 – dacă $I < 0$, taskul în care se execută primitiva, iese din rândul taskurilor rulabile, se blochează iar indexul său este înscris în coada de aşteptare *C.* În aceste împrejurări se provoacă un proces de comutare care determină trecerea în rulare (*execuţie fizică*) a altui task.

[3] Indivizibilitatea are în vedere faptul că cele două funcţii nu pot fi întrerupte.
[4] **F**irst **I**nput **F**irst **O**utput

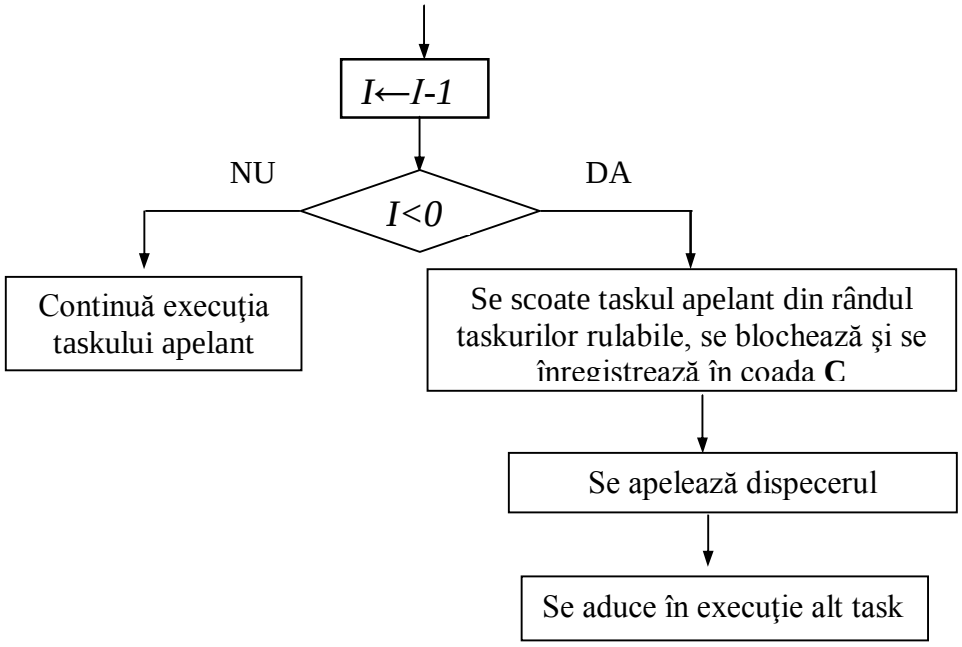

Fig. 1. . Execuţia primitivei *P(S)*.

Primitiva *V(S)* presupune realizarea următoarelor operaţii (ilustrate în organigrama din figura 2.):

1 - *I←I+1;*

2 – dacă $I \le 0$, se deblochează taskul aflat pe prima poziţie în coada de aşteptare la semafor şi se înscrie în rândul taskurilor rulabile, după care se face apel la dispecer, după care continuă execuţia taskului apelant ;

3 – dacă *I > 0,* taskul în care se execută primitiva, îşi continuă execuţia.

În timpul execuţiei primitivei *V(S),* dacă în urma incrementării rezultă *I≤0,* după deblocarea taskului aflat pe prima poziţie în coada **C** se dă controlul dispecerului. Acesta decide funcţie de prioritate dacă aduce sau nu în execuţie taskul deblocat[5].

Din definirea primitivelor *P(S)* şi *V(S)* rezultă următoarele precizări referitoare la valorile variabilei *I* aferente semaforului *(S)*:

- dacă *I > 0* , atunci aceasta reprezintă numărul de taskuri care pot executa primitiva *P(S)* fără a se bloca, presupunând că între timp nu se execută nici o primitivă *V(S);*

[5] În figura 2.21 este surprinsă situaţia în care continuă execuţia taskului apelant, deci taskul deblocat este adus în execuţie logică.

- dacă $I \leq 0$, atunci $|I|$ reprezintă numărul de taskuri blocate la semaforul S şi înregistrate în coada **C.**

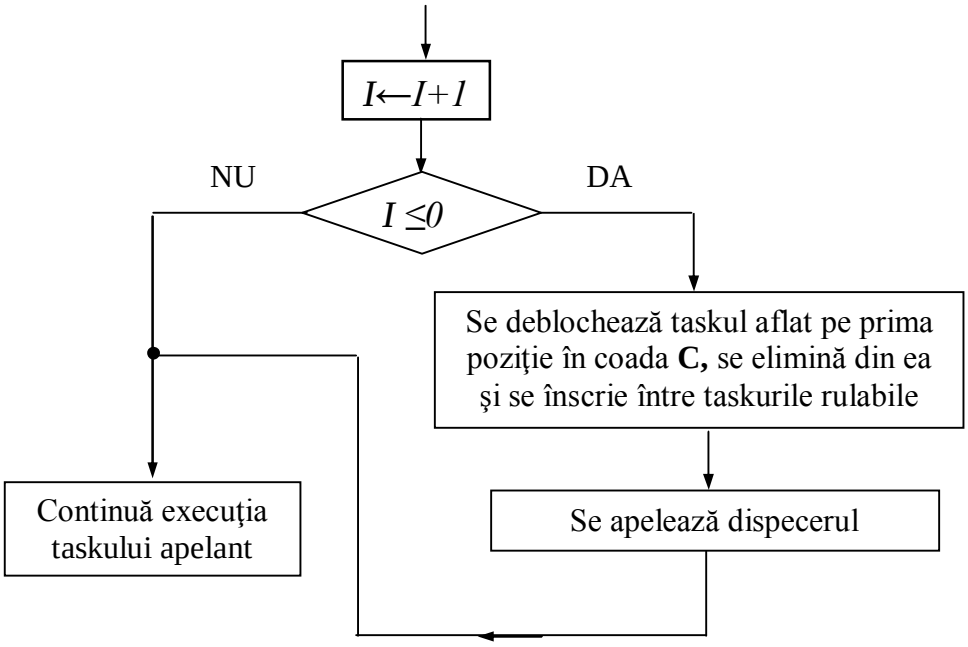

Fig. 3. Execuţia primitivei *V(S)*.

Excluderea mutuală cu semafoare presupune utilizarea unui singur semafor binar, pe care îl vom nota *SEMEX,* care se iniţializează cu valoarea *1*. În fiecare task care trebuie să se excludă mutual, înainte de intrarea în secţiunea critică se execută o directivă *P(SEMEX),* iar după ieşirea din secţiune o directivă *V(SEMEX).*

Este clar că în intervalul de timp în care un task se află în propria secţiune critică referitoare la o anumită resursă *SEMEX=0.* În aceste condiţii orice alt task ce va dori să acceadă în propria secţiune critică referitoare la aceiaşi resursă, va executa directiva *P(SEMEX),* ceea ce va determina blocarea sa întrucât după decrementare *SEMEX =-1*.

La ieşirea din secţiunea critică execuţia directivei *V(SEMEX)* va determina *SEMEX=0* şi conform celor precizate anterior , taskul aflat în prima poziţie în coada **C** aferentă semaforului *S* , va fi deblocat[6] şi va putea pătrunde la rândul său în propria secţiune critică.

[6] Pentru ca soluţia să fie funcţională, semaforul *SEMEX* nu trebuie aservit altor scopuri în afara excluderii mutuale.

Ca exemplu în figura 4 se prezintă schemele logice aferente excluderii mutuale cu semafoare a două taskuri *T1, T2*.. Se observă că cele două taskuri se execută la intervale prestabilite *Δt_ex_T1, Δt_ex_T1*.

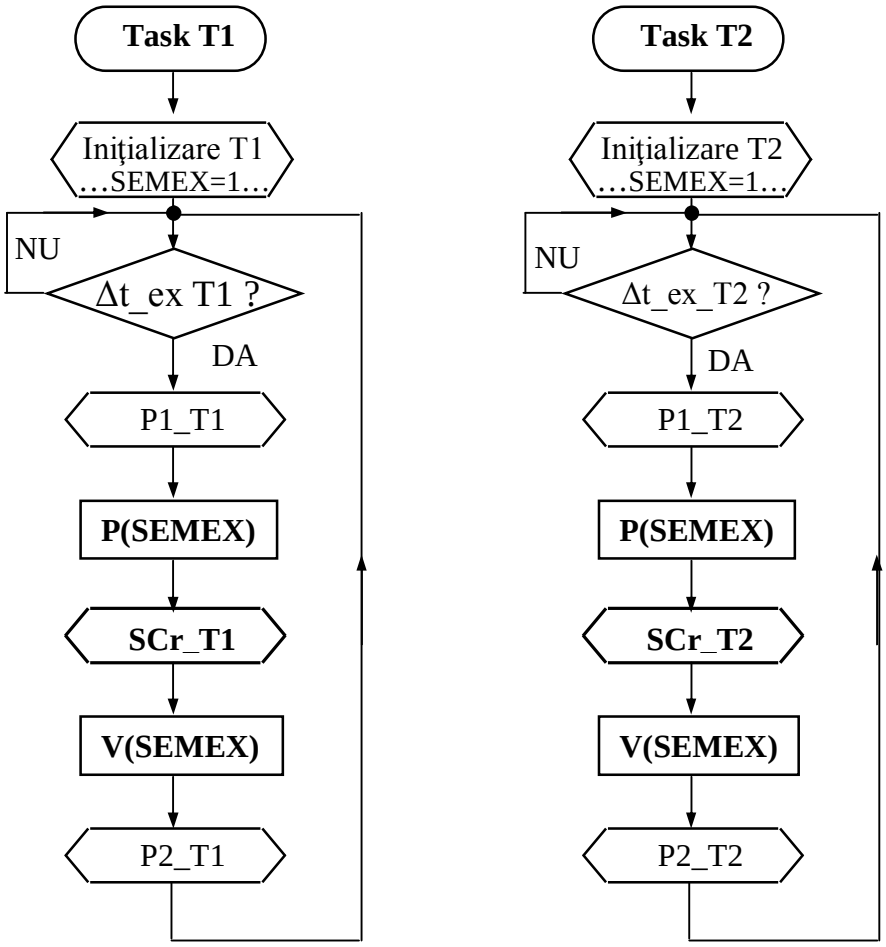

Fig. 4. Excluderea mutuală cu semafoare:
P1_T1, …,P2_T2 proceduri ale taskurilor T1 şi T2.

Având în vedere că *SEMEX* este iniţializat cu valoarea *1,* dacă de exemplu pentru *T1*se impune accesul în secţiunea sa critică, acest lucru va fi posibil întrucât directiva *P(SEMEX)* nu va bloca *Taskul T1*. Dacă în timp ce *T1* se află în secţiunea sa critică *T2* va ajunge la iminenţa intrării în secţiunea sa critică, acest deziderat nu se va putea realiza întrucât directiva *P(SEMEX)* va determina blocarea *Taskului T2* la semaforul *SEMEX*. Deblocarea se va produce când *T1* va termina de executat secţiunea sa critică după care va executa directiva *V(SEMEX).* Ca urmare a acestei directive *SEMEX* va căpăta valoarea *0 (zero)* şi *T2* va deveni rulabil urmând a fi planificat de dispecer pentru execuţia secţiunii sale critice.

Din cele prezentate rezultă că se realizează excluderea mutuală, respectiv un singur task se poate găsi la un moment dat în propria sa secţiune critică.

3. Excluderea mutuală realizată cu mesaje şi cutii poştale

O cutie poştală *(CP)* reprezintă o structură al cărui element central este un tampon circular gestionat conform principiului *FIFO*. Într-o *CP* taskurile pot depune mesaje, accesibile oricărui task. În mod normal o *CP* se specifică prin numărul de poziţii ale tamponului circular şi prin lungimea admisă pentru un mesaj.

Mesajele reprezintă volume de date transferate între taskuri pe parcursul evoluţiei lor. Mesajele pot fi transmise direct între taskuri (caz în care se numesc *mesaje de trecere)* su prin intermediul cutiilor poştale. Din punct de vedere al conţinutului mesajele pot fi cu conţinut *fix* sau *variabil.* Cele cu conţinut fix se numesc *mesaje simbolice* iar celelalte *mesaje informaţionale.* În ceea ce priveşte formatul şi acesta poate fi *fix* sau *variabil.* În mod normal mesajele simbolice sunt mesaje cu format fix.

O funcţie de depunere *(PUT)* a unui mesaj într-o anume *CP* trebuie să conţină un pointer la variabila al cărui conţinut se depune, în timp ce o funcţie de extragere *(GET)* a unui mesaj dintr-o anume *CP* va trebui să conţină un pointer la variabila în care se depune mesajul extras.

Cutiile poştale trebuie astfel gestionate încât să se producă blocarea taskurilor pe *operaţii de depunere* , atunci când *CP* este plină sau pe *operaţii de extragere* atunci când *CP* este vidă.

Excluderea mutuală cu *cutii poştale* presupune utilizarea unei singure *CP* pe care o vom nota *CPEX* în care primul task ce se iniţializează depune un mesaj simbolic, notat *MESEX*. Înainte de intrarea în secţiunea critică fiecare task care trebuie să se excludă mutual va extrage *MESEX* din *CPEX*, iar după ieşirea din secţiunea critică îl va redepune.

Rezultă că în intervalul de timp în care un task se află în secţiunea sa critică referitoare la o anumită resursă cutia poştală *CPEX* este vidă. În aceste condiţii orice alt task ce va dori să acceadă în propria secţiune critică referitoare la aceiaşi resursă, se va bloca la execuţia funcţiei de preluare a mesajului simbolic. Deblocarea[7] se va produce în momentul în care *CPEX* va conţine din nou mesajul simbolic *MESEX*

[7] Utilizarea căsuţei poştală *CPEX* şi a mesajului *MESEX* trebuie să fie numai în excludere mutuală. utilizată numai la implementarea excluderii mutuale.

În figura 5 se prezintă schemele logice aferente excluderii mutuale cu utilizarea cutiilor poştale a două taskuri, *T1*şi *T2*. Ca şi în cazurile precedente cele două taskuri sunt sincronizate cu timpul, intervalele de execuţie fiind *Δt_ex_T1, Δt_ex_T1*.

Având în vedere că la iniţializare cutia poştală *CPEX* va conţine mesajul simbolic *MESEX*, dacă de exemplu pentru *T1*se impune accesul în secţiunea sa critică, acest lucru va fi posibil întrucât prin funcţia de extragere *GET* va putea fi preluat mesajul *MESEX*. Dacă în timp ce *T1* se află în secţiunea sa critică *T2* va ajunge la iminenţa intrării în secţiunea sa critică, acest deziderat nu se va putea realiza întrucât funcţia *GET* va determina *Taskului T2* la cutia poştală *CPEX*. Deblocarea se va produce când *T1* va termina de executat secţiunea sa critică după care va executa funcţia *PUT* de depunere a *MESEX* în *CPEX*.

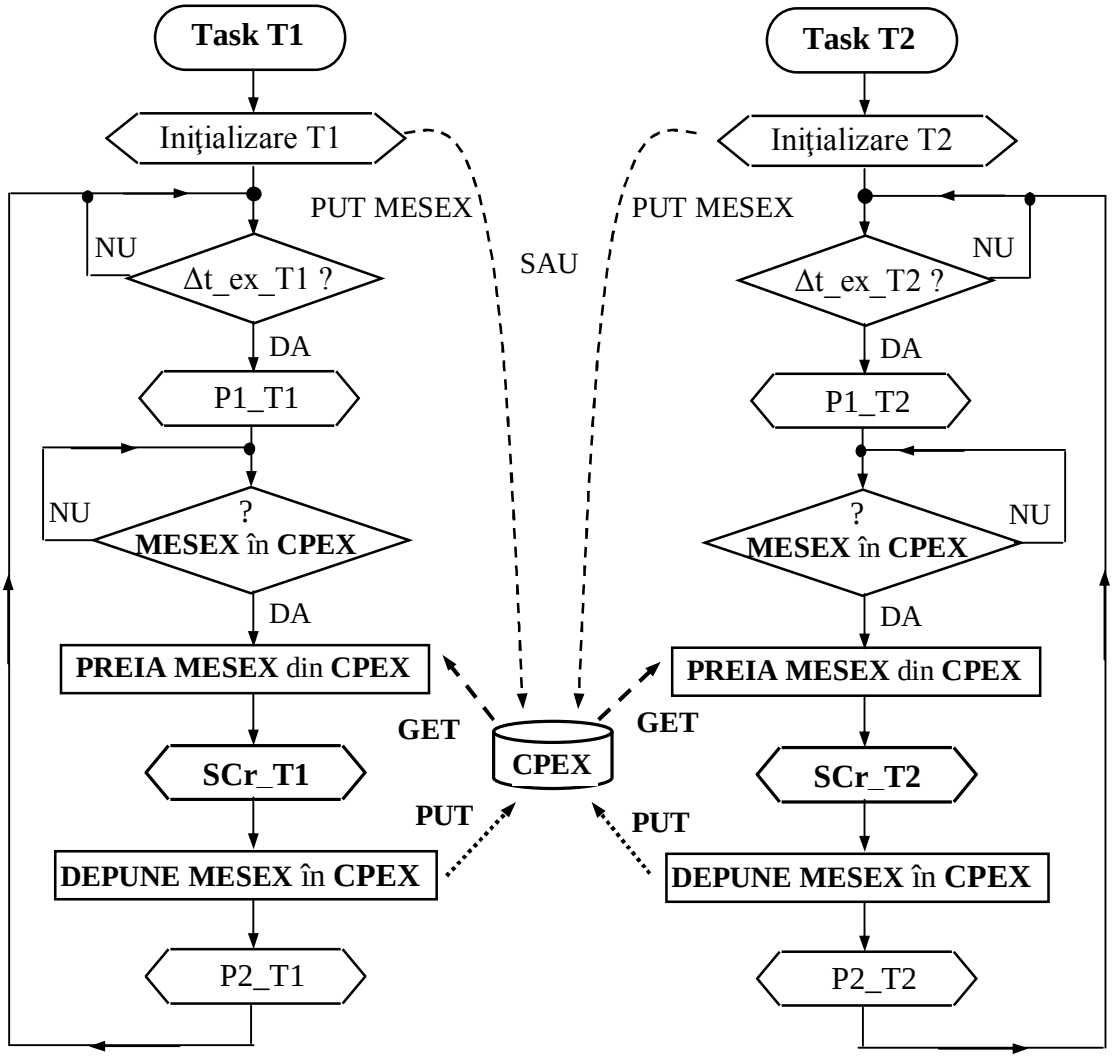

Fig. 5. Excluderea mutuală cu cutii poştale şi mesaje:
P1_T1, …,P2 _T2 proceduri ale taskurilor T1 şi T2.

În urma acestei depuneri *T2* va deveni executabil urmând a fi planificat de dispecer pentru execuţia secţiunii sale critice.

Din cele prezentate rezultă că şi prin utilizarea *cutiilor poştale şi a mesajelor* se poate realiza excluderea mutuală, respectiv se poate asigura prezenţa unui singur task la un moment dat în propria sa secţiune critică.

3. Sincronizarea taskurilor cu utilizarea semafoarelor

De regulă, în evoluţia lor, taskurile unei aplicaţii trebuie să se supună unor relaţii de ordine care să le asigure o anumită *succesiune temporală.*

De exemplu într-o aplicaţie de conducere în timp real un task efectuează achiziţia datelor din proces şi le depune într-un buffer de unde vor fi preluate de taskurile utilizator (reglare, supraveghere, monitorizare etc.). În această situaţie taskurile beneficiare trebuie să aştepte încărcarea buffer-ului, după care în cadrul unei secţiuni critice vor prelua datele.

Sincronizarea taskurilor reprezintă procesul de punerea a unui task în relaţie cu *alt task, cu timpul sau cu un eveniment extern.* Două taskuri se consideră sincronizate dacă se pot stabili relaţii predictibile între anumite momente ale desfăşurării lor. Sensul general al *sincronizării* este acela de coordonare în timp, decorelare.

În mod obişnuit sincronizarea se poate face:

- funcţie de producerea unui eveniment extern taskului;
- funcţie de realizarea unei condiţii de timp.

Sincronizarea cu timpul poate fi tratată ca o sincronizare cu evenimente externe dacă se consideră impulsurile ceasului de timp real ca fiind astfel de evenimente. Pentru realizarea sincronizării sunt necesare funcţii specifice de tratare şi anume:

- *wait* – aşteptare până la producerea evenimentului;
- *signal* – anunţare că evenimentul s-a produs.

Metodele şi mecanismele utilizate pentru realizarea sincronizării se deosebesc sub mai multe aspecte şi anume:

- *natura sincronizării* (cu evenimente externe sau cu timpul);
- *momentul sincronizării (*cu începutul, zona mediană sau sfârşitul unui task):
- *implementarea sincronizării* (prin facilităţi ale limbajelor de programare, ale limbajelor de comandă sau ale monitoarelor).

Din punctul de vedere al sincronizării cu timpul pot exista două situaţii, ilustrate în figura 6, şi anume:

- taskul aşteaptă expirarea unui interval de timp Δt după care se execută (figura 6 a);
- taskul se execută în interiorul cuantei de timp Δt (figura 6 b);.

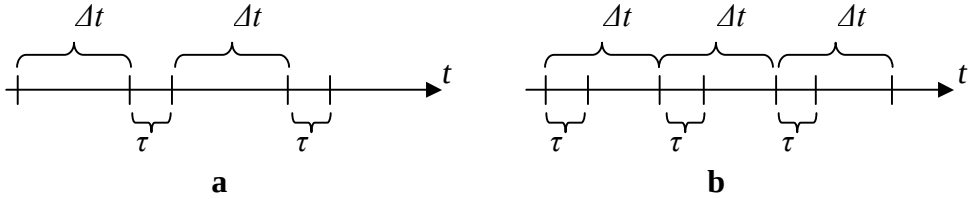

Fig. 6. Sincronizarea cu timpul:
a – intervalul de execuţie τ în afara intervalului de sincronizare Δt;
b – intervalul de execuţie τ în interiorul intervalului de sincronizare Δt.

Se va exemplifica utilizarea semafoarelor pentru un sistem de două taskuri *T1* şi *T2* care trebuie să se sincronizeze reciproc, în fiecare task fiind definit câte un punct de sincronizare *PS1* pentru *T1,* respectiv *PS2* pentru *T2.* Sincronizarea trebuie astfel realizată încât taskul *Ti* să nu poată trece de punctul său de sincronizare *PSi* până când celălalt task *Tj* nu a ajuns în punctul său de sincronizare *PSj* ($i, j \in \{0,1\}$).

Din analiza figurii 7, în care se prezintă sincronizarea cu semafoare, rezultă că punctele de sincronizare se află între procedurile *P1* şi *P2* ale fiecărui task. Cu alte cuvinte un task nu poate executa propria procedură *P2* până când celălalt task nu şi-a executat procedura *P1.*

Sincronizarea se realizează cu ajutorul a doua semafoare binare notate în figura 7 cu *SEMSYNC1* şi *SEMSYNC2* care în taskurile *T1* şi *T2* se iniţializează cu valorile ***1*** respectiv ***0.*** În punctul de sincronizare *PSi* al taskului *Ti* se execută secvenţa *P(SEMSYNCi), V(SEMSYNCj)* cu $i, j \in \{0,1\}$.

Dacă de exemplu *T2* ajunge în *PS2* înainte ca *T1* să îşi execute procedura *P1,* se va bloca întrucât *SEMSYNC2=0.* Taskul se va debloca după ce *T1* depăşeşte *PS1* deoarece va executa directiva *V(SEMSYNC2).*

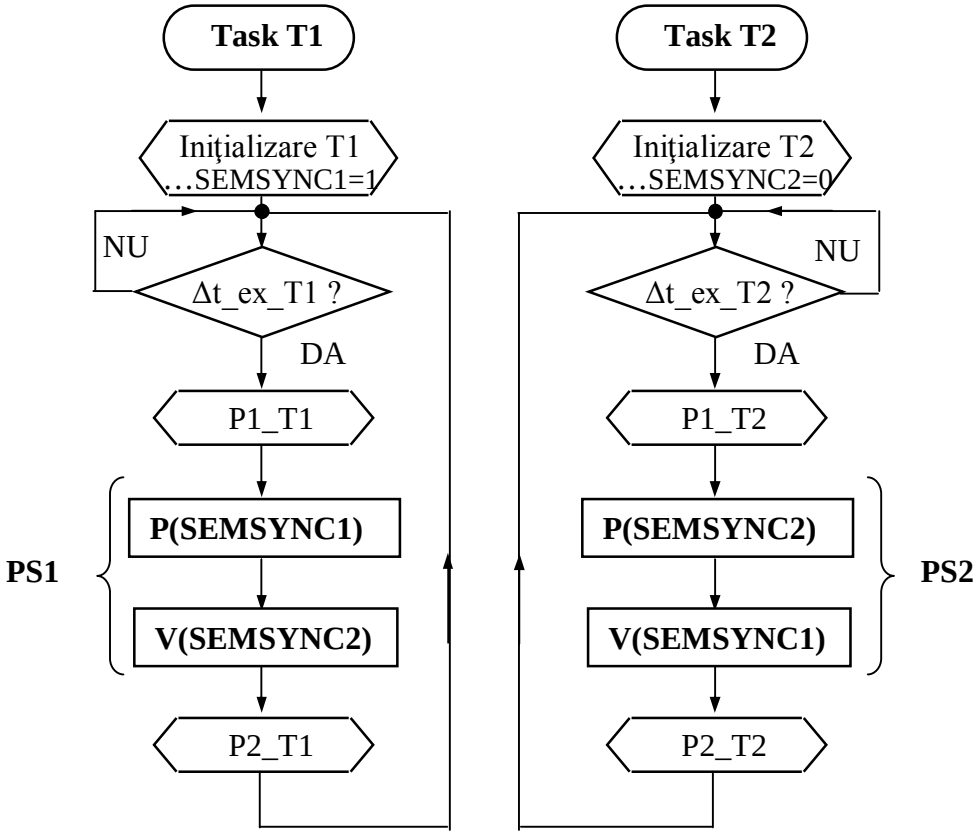

Fig. 7. Utilizarea semafoarelor pentru sincronizarea a două taskuri: P1_T1, …,P2_T2 proceduri ale taskurilor T1 şi T2.

În figura 8 este ilustrată utilizarea sincronizării cu un eveniment extern. Taskul *T1* trebuie să îşi execute procedura *P2* numai după producerea evenimentului extern *EVEXT*. În acest scop se utilizează semaforul de sincronizare *SYMSYNC* care se iniţializează cu valoarea ***0.*** Asupra acestui semafor se execută o directivă de tip *V* în taskul *T0* care supraveghează producerea evenimentului.

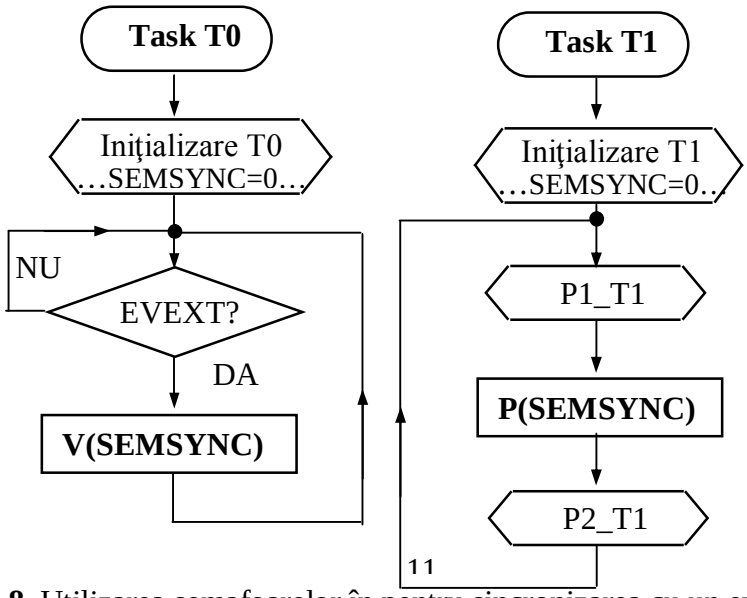

Fig. 8. Utilizarea semafoarelor în pentru sincronizarea cu un eveniment extern: P1_T1, …,P2_T1 proceduri ale taskului T1.

Semafoarele pot fi utilizate şi în sincronizarea cu timpul, în figura 9 fiind prezentat un exemplu în acest sens. Taskul *T1* trebuie să se execute la intervale de timp *Δt_ex_T1* . În acest scop se construieşte taskul *T0* cu rol de planificator. Acesta execută o aşteptare temporizată la semaforul de sincronizare *SYMSYNC* iniţializat cu valoarea ***0.*** Aşteptarea temporizată presupune execuţia directivei *V(SEMSYNC)* la expirarea intervalului *Δt_ex_T1*. Această directivă va debloca taskul *T1,* care între momentele de execuţie aşteaptă pe o funcţie *P* la acelaşi semafor, rezultatul fiind acela că taskul *T1* se va executa cu periodicitatea *Δt_ex_T1*[8] .

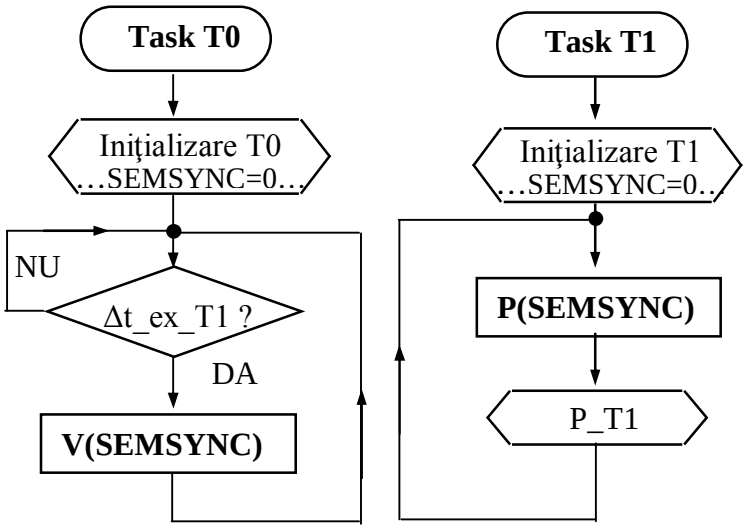

Fig. 9. Utilizarea semafoarelor în pentru sincronizarea cu timpul:
P_T1 procedură a taskului T1.

4. Sincronizarea realizată cu mesaje şi cutii poştale

Cutiile poştale şi mesajele pot fi utilizate atât pentru implementarea sincronizării cu timpul, cât şi cu evenimente externe.

În cazul sincronizării cu timpul se utilizează de regulă *aşteptarea temporizată* la o cutie poştală vidă. Mesajele implicate sunt de regulă *mesaje simbolice,* respectiv mesaje cu format şi conţinut fix. În continuare vor fi prezentate două soluţii de sincronizare bazate pe cutii poştale şi mesaje.

O primă soluţie, ilustrată în figura 10 presupune existenţa a două taskuri *T1- cu rol de planificator* şi *T2 – care trebuie să se execute la intervale Δt.* Soluţia implică prezenţa a trei cutii poştale *C0, C1, C2* cu următoarele funcţii:

[8] Timpul de execuţie al taskului *T1* este inclus în acest interval.

- *C0* – CP destinată aşteptării temporizate;
- *C1* – CP destinată transferului mesajului de sincronizare *MES_SYNC;*
- *C2* – CP destinată transferului mesajului de confirmare *MES_CONF.*

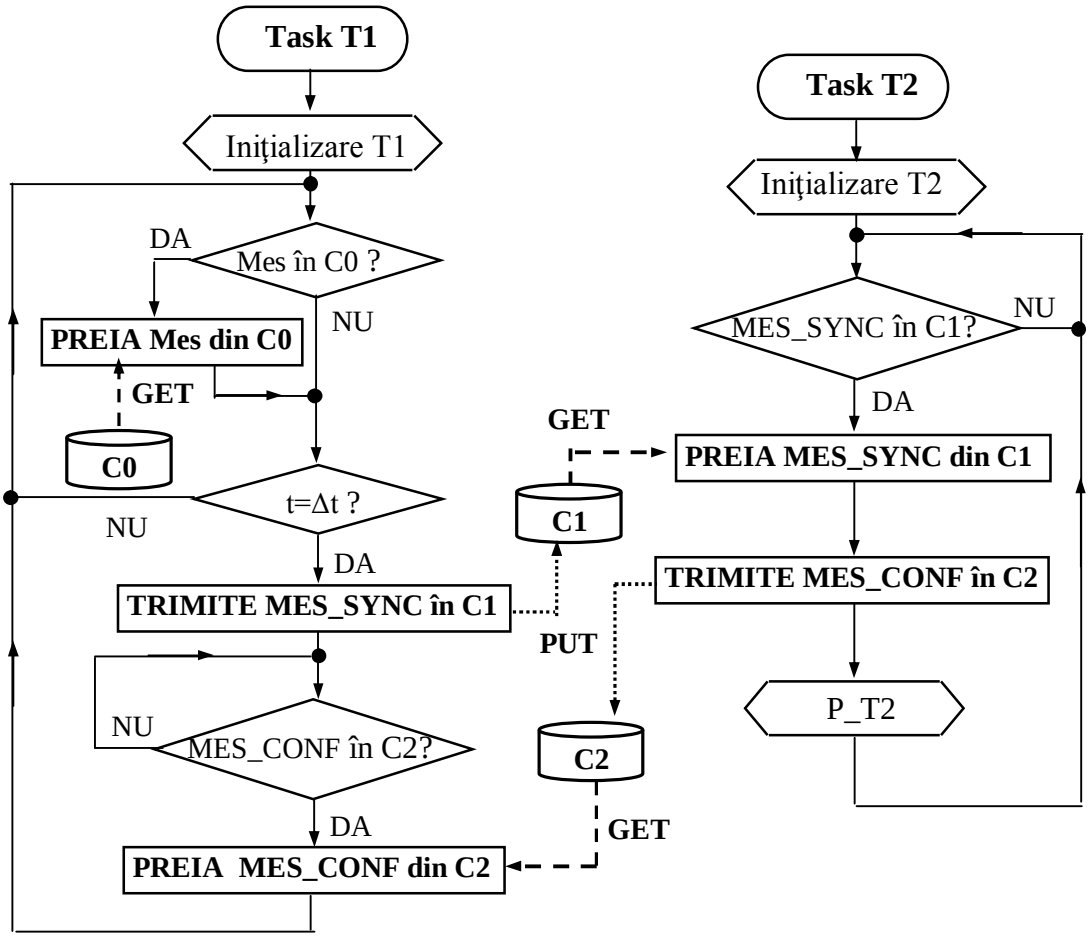

Fig. 10. Utilizarea cutiilor poştale pentru sincronizarea cu timpul:
C0, C1, C2 – cutii poştale; P T2 procedură ale taskului T2.

Taskul *T1* realizează următoarea secvenţă de operaţii:

1.1 – aşteaptă un interval de timp *Δt (intervalul de sincronizare)* la cutia poştală vidă *C0;*

1.2 – la expirarea acestui intervalul transmite în cutia poştală *C1* mesajul de sincronizare *MES_SYNC;*

1.3 – aşteaptă şi preia din cutia poştală *C2* mesajul de confirmare *MES_CONF;*

1.4 – se revine la pasul 1.1 , intrându-se din nou în starea de aşteptare temporizată.

Mesajul de confirmare *MES_CONF* este un mesaj cu formă fixă care informează taskul planificator *T1* în legătură cu funcţionalitatea taskului *T2*, care trebuie să se execute temporizat.

În ceea ce priveşte taskul *T2* , acesta execută următoarea secvenţă:

2.1 – aşteaptă şi preia din cutia poştală *C1* mesajul de sincronizare *MES_SYNC;*

2.2 - transmite în cutia poştală *C2* mesajul de confirmare *MES_CONF;*

2.3 – execută procedura *P-T2.*

Din secvenţa de mai sus rezultă faptul că execuţia taskului *T2* se produce în interiorul cuantei de timp *Δt* , aspect evidenţiat în figura 11.

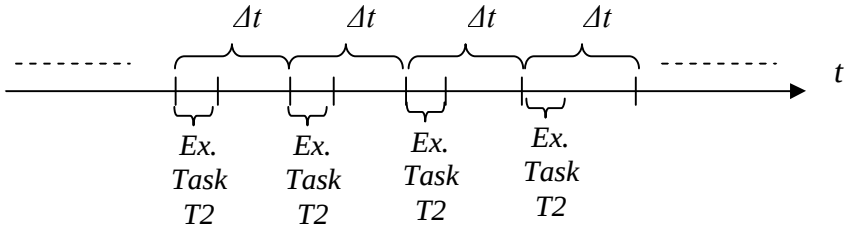

Fig. 11. Sincronizarea cu timpul: execuţia taskului *T2* se face în interiorul cuantei de timp *Δt.*

Este posibilă execuţia taskului *T2* în afara cuantei *Δt,* conform reprezentării din figura 12.

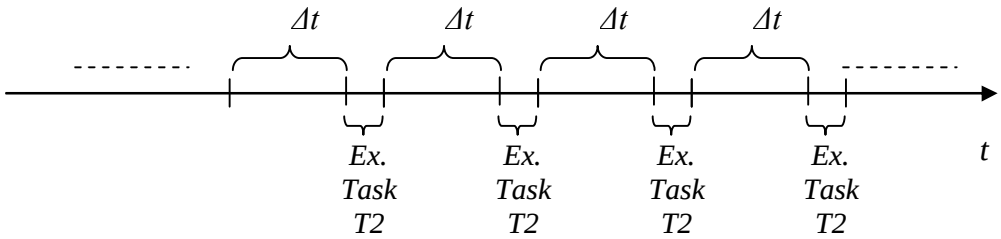

Fig. 12. Sincronizarea cu timpul: execuţia taskului *T2* se face în exteriorul cuantei de timp *Δt.*

În această situaţie nu se mai impune existenţa taskului planificator *T1*. Taskul *T2* aşteaptă la cutia poştală vidă *C0* un interval de timp *Δt,* deblocarea producându-se la expirarea acestei cuante de timp. În figura 13 se prezintă structura adaptată a taskului *T2* pentru acest tip de sincronizare.

Se observă că nu mai sunt necesare mesaje de sincronizare şi de confirmare. Dacă se doreşte monitorizarea funcţionalităţii taskului *T2,* atunci se menţine taskul *T1* care va primeşte la fiecare execuţie a lui *T2* câte un mesaj de confirmare.

Procedeul sincronizării cu timpul, în condiţiile existenţei unui task planificator poate extins şi la implementarea sincronizării cu un eveniment extern, corespunzător reprezentării din figura 14.

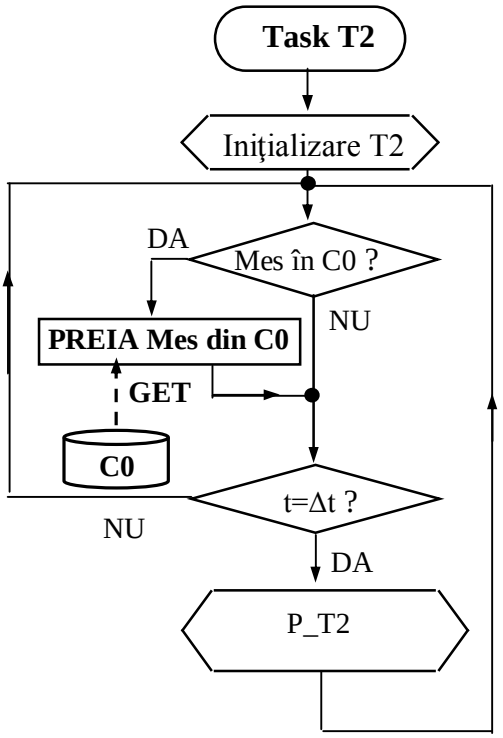

Fig. 2.33. Utilizarea unei cutii poştale vide pentru sincronizarea cu timpul: C0 – cutie poştală; P_T2 procedură ale taskului T2.

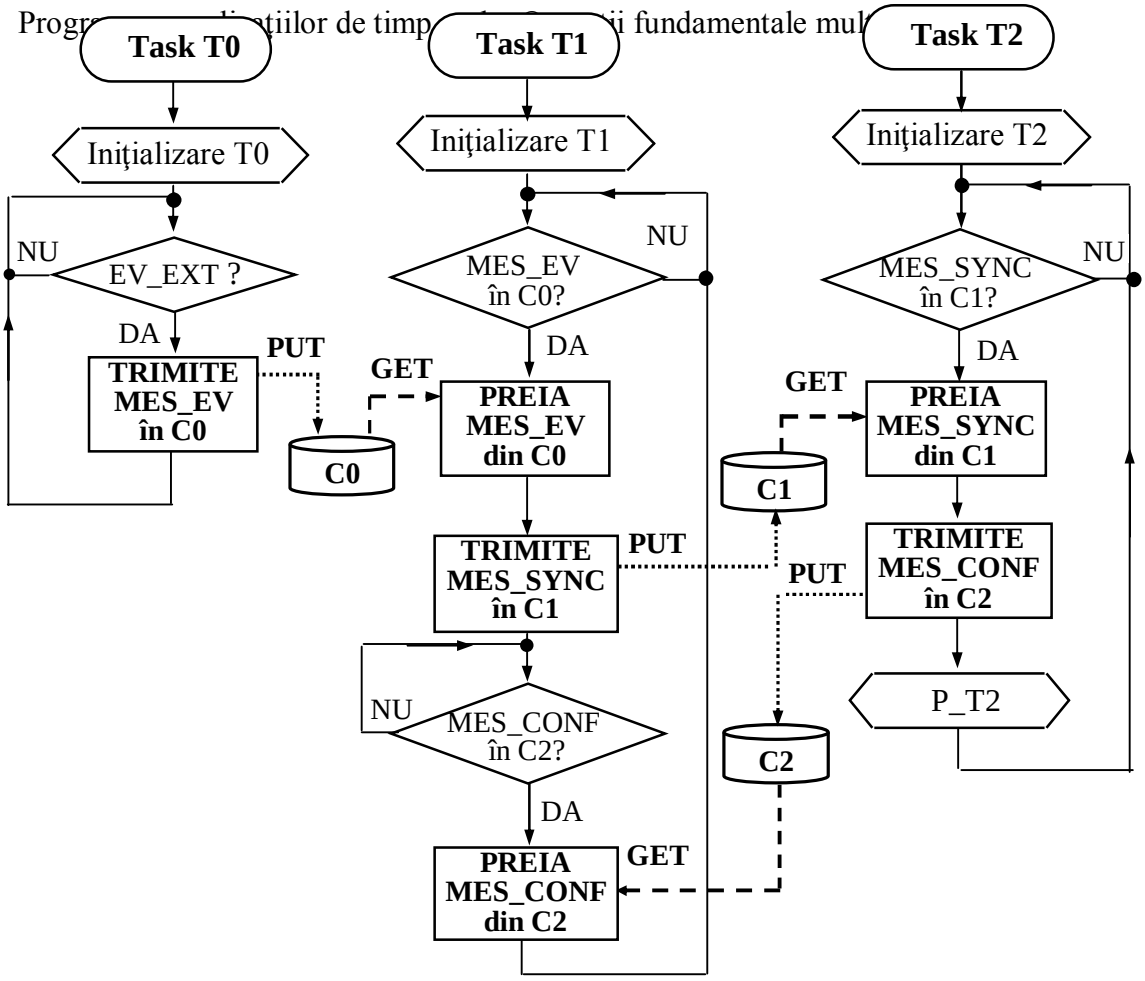

Fig. 14. Utilizarea cutiilor poştale pentru sincronizarea cu evenimente externe: C0, C1, C2 – cutii poştale; P_T2 procedură ale taskului T2.

După cum se observă, înainte de producerea evenimentului *EV_EXT* taskurile *T0, T1, T2* sunt blocate după cum urmează:

- *T0* – în aşteptarea producerii evenimentului *EV_EXT;*
- *T1* – la cutia poştală *C0* în aşteptarea mesajului *MES_EV* care confirmă producerea evenimentului;
- *T2* – la cutia poştală *C1* în aşteptarea mesajului de sincronizare *MES_SYNC.*

Este important de subliniat faptul că după primirea mesajului de sincronizare, *T2* trebuie să depună în cutia poştală *C2* mesajul de confirmare *MES_CONF.*

Ca şi în cazul sincronizării cu timpul, taskul *T1* poate fi eliminat în situaţia în care nu se doreşte monitorizarea funcţionalităţii taskului *T2*, soluţia principială fiind prezentată în figura 15.

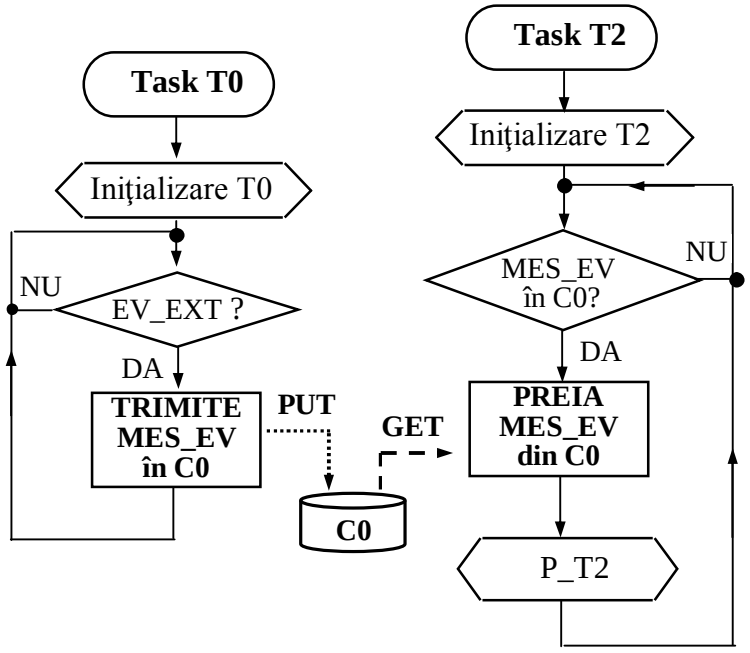

Fig. 15. Utilizarea unei cutiilor poştale pentru sincronizarea cu evenimente externe: C0, – cutie poştale; P_T2 procedură ale taskului T2.

După cum se observă între momentele producerii evenimentului extern *EV_EXT* taskurile *T0* şi *T2* sunt blocate după cum urmează:

- *T0* – în aşteptarea producerii evenimentului *EV_EXT;*
- *T2* – la cutia poştală *C0* în aşteptarea mesajului *MES_EV* referitor la producerea evenimentului extern.

În aceste condiţii taskul *T2* este deblocat în momentul în care în cutia poştală *C0* este depus *MES_EV* asociat producerii evenimentului extern.

5. Utilizarea cutiilor poştale şi mesajelor în comunicare

După cum s-a arătat procesarea datelor în regim concurent multitasking implică pe lângă utilizarea de resurse în comun şi schimbul de informaţie între taskuri concretizat în operaţia de *comunicare*. Uzual taskurile transferă informaţia sub formă de mesaje care, după cum s-a văzut la prezentarea excluderii mutuale pot fi cu conţinut *variabil (mesaje informaţionale)* sau *fix (mesaje simbolice).*

În mod obişnuit comunicarea se implementează prin mecanismul *producător-consumator,* taskurile putând fi din acest punct de vedere

- taskuri de tip *producător;*
- taskuri de tip *consumator.*

Pentru facilitarea comunicării sistemele de operare trebuie să pună la dispoziţie instrumente specifice, unul dintre acestea fiind conducta *(pipe).* O conductă reprezintă un tampon unidirecţional gestionat conform principiului *FIFO,* în care un task producător depune mesaje pe care le preia un task consumator.

Mecanismul de comunicare prin conductă trebuie să asigure blocarea taskului *producător* care ajunge în faţa unei operaţii de scriere şi conducta este plină. Acelaşi mecanism va trebui să blocheze un task de tip consumator aflat în situaţia de preluare a unui mesaj (citire) dintr-o conductă goală.

În situaţia în care două taskuri au atât rol de producător, cât şi de consumator vor trebui utilizate două conducte, situaţie evidenţiată în figura 16.

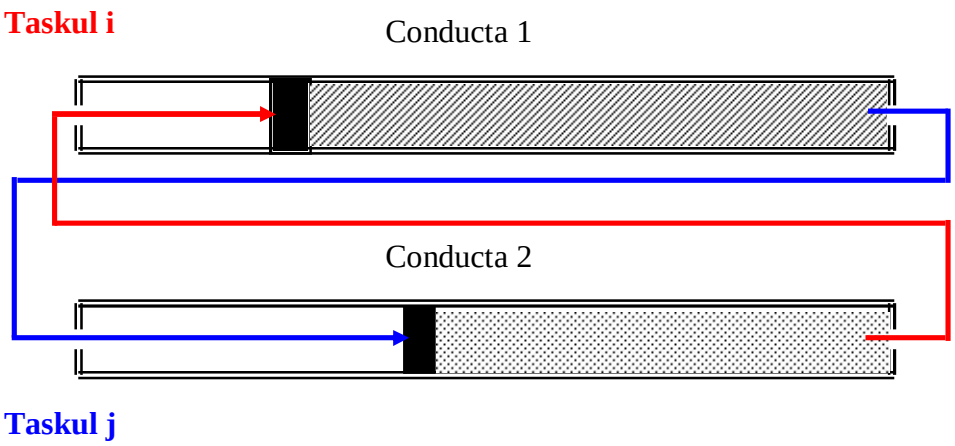

Fig. 16. Ilustrarea comunicării prin conducte între două taskuri.

După cum se observă *Conducta 1* este deschisă la scriere pentru *taskul i* şi la citire pentru *taskul j,* cu alte cuvinte cele două taskuri îndeplinesc rolurile de *producător* respectiv *consumator.* În ceea ce priveşte *Conducta 2,* aceasta este deschisă la scriere pentru *taskul j* şi la citire pentru *taskul i,* rolurile celor două taskuri fiind inversate respectiv *taskul i – consumator, taskul j – producător.*

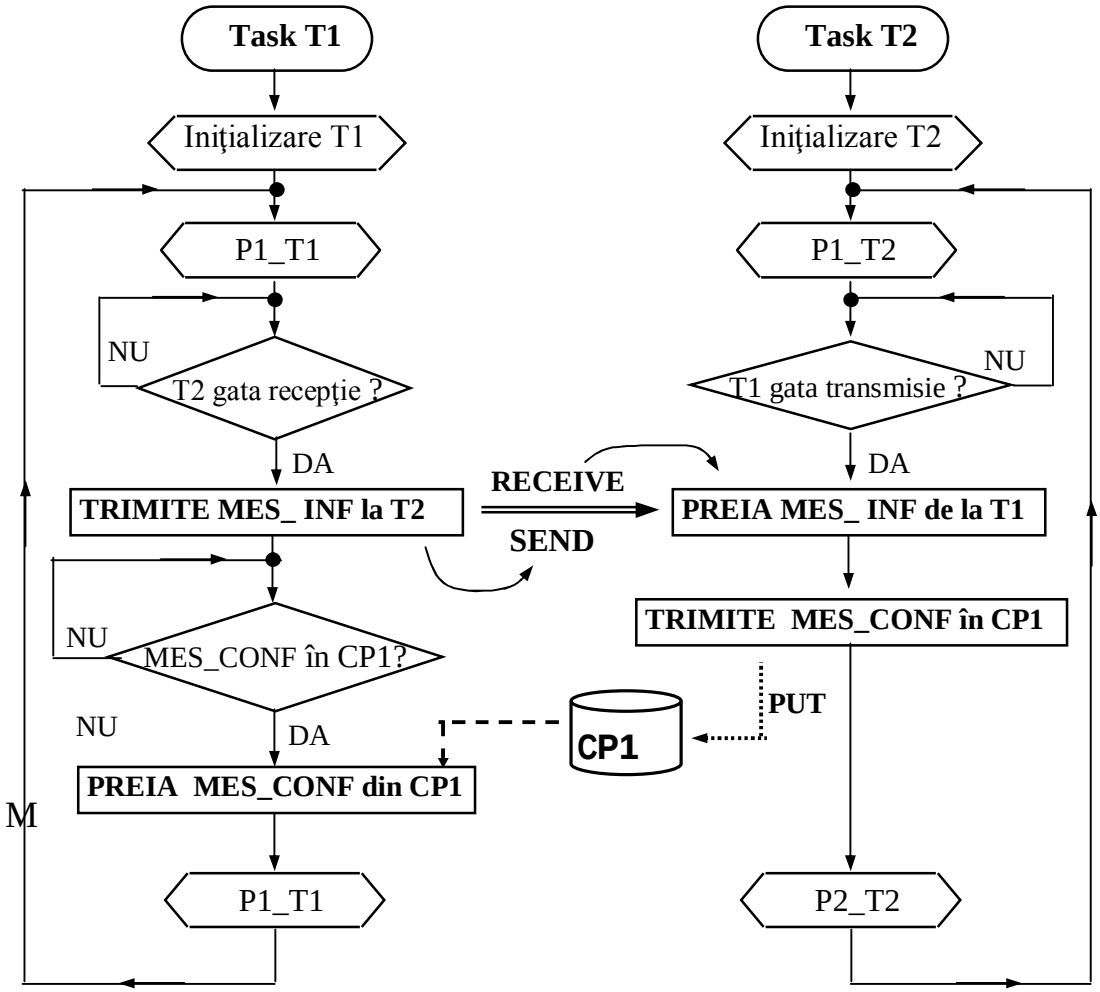

Fig. 17. Utilizarea mesajelor de trecere şi a cutiilor poştale în comunicare.

Mesajele de trecere şi cutiile poştale reprezintă alte două mijloace destinate asigurării comunicării dintre taskuri. Mesajele se trimit direct de la un taskul emiţător către cel receptor, în timp ce cutiile poştale reprezintă facilităţi de comunicare care pot fi utilizate de către toate taskurile aferente aplicaţiei de timp real.

În figura 17 se prezintă un exemplu de comunicare între două taskuri în care se utilizează ambele modalităţi. Mesajul informaţional *MES_INF* este transmis de *Taskul T1* sub forma unui mesaj de trecere în timp ce *Taskul T2* va trimite un mesajul simbolic de confirmare *MES_CONF* prin intermediul cutiei poştale *CP1*.

Este important de subliniat faptul că transmiterea unui mesaj de trecere este posibilă dacă taskul receptor este gata să îl primească. În ceea ce priveşte receptorul acesta se va bloca în aşteptarea mesajului dacă taskul emiţător nu este gata să îl transmită.

În ceea ce priveşte mesajul de confirmare, *Taskul T1* se blochează până la depunerea acestuia în cutia poştală *CP1*.

Sisteme de interfaţă proces – calculator

Necesitatea interconectării subsistemelor cu caracteristici complet diferite impune utilizarea unor dispozitive care să le compatibilizeze. Această compatibilizare trebuie să asigure transferul informaţional între sistemele conectate, care devine posibil în condiţiile în care dispozitivele de interconectare prezintă caracteristici comune ambelor subsisteme.

În cadrul sistemelor numerice de conducere a proceselor, compatibilizarea este asigurată de către *sistemul de interfaţă proces-calculator (SIPC)* ale cărui caracteristici vor fi prezentate în cele ce urmează.

1. Structura şi funcţiile sistemului de interfaţă cu procesul

Necesitatea existenţei *SIPC* apare evidentă datorită incompatibilităţii totale între mărimile aferente procesului şi cele cu care operează echipamentul numeric de conducere. Practic *SIPC* asigură posibilitatea conectării calculatorului la echipamentele periferice de proces reprezentate de *traductoare şi elemente de execuţie.*

La nivelul *SIPC* se consideră că are loc o transmisie bilaterală de informaţie, între proces şi calculator, care funcţie de sensul transmisiei se pot constitui alternativ în emiţătoare respectiv receptoare de informaţie. Transmisia semnalelor se poate face prin cablu, radio, fibră optică, etc. În cazul transmisiei prin cablu, aceasta se poate face *serie sau paralel*, durata fiind determinată, printre altele, de: timpii de trecere, de identificare a receptorului şi de acceptare a transmisiei. Viteza de transmisie a datelor este influenţată şi de tipul semnalelor de sincronizare.

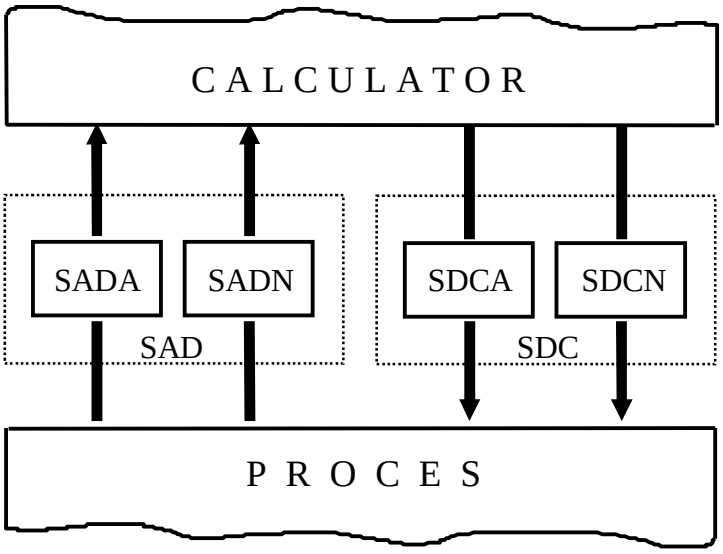

Fig. 1. Structura generalăa unui SIPC.

Structura generală a unui *SIPC* este de forma celei prezentate în figura 1, în care se remarcă prezenţa a două subsisteme şi anume:

- subsistemul de achiziţie a datelor *(SAD);*

- subsistemul de distribuţie a comenzilor *(SDC).*

◆ *SAD*, care cuprinde subsistemele de achiziţie a datelor analogice *(SADA)* şi numerice *(SADN)*, asigură *preluarea semnalelor din proces, adaptarea lor în vederea prelucrării numerice şi transmisia la calculator*.

◆ *SDC*, care cuprinde subsistemele de distribuţie a comenzilor în forma analogică *(SDCA)* şi numerică *(SDCN), asigură transformarea informaţiei furnizate de calculator în semnale specifice elementelor de execuţie (analogice sau numerice) şi transmisia către acestea.*

Gestionarea transferului de informaţie între perifericele de proces şi echipamentul numeric de conducere este realizată de către o *bloc de comandă* aferent *SIPC.*

În momentul actual se constată tendinţa de *migrare* a tehnicii numerice către perifericele de proces aspect materializat de apariţia aşa numitelor *traductoare şi elemente de execuţie inteligente*. Această deplasare nu schimbă însă fondul atribuţiilor *SIPC* deoarece subsistemele aferente acestuia se regăsesc în structurile traductoarelor şi elementelor de execuţie. Soluţia, tentantă la prima vedere, prezintă inconvenientul costurilor, încă destul de ridicate pentru aceste tipuri de traductoare şi de elemente de execuţie.

O situaţie frecvent întâlnită este aceea în care se menţin traductoare şi elemente de execuţie analogice dar se utilizează *regulatoare numerice*. În acest caz regulatoarele conţin interfeţe de proces cu structuri apropiate de cele ce se vor prezentata în continuare.

2. Subsistemul de achiziţie a datelor analogice (SADA)

Un volum important din informaţia privind starea proceselor tehnologice este transmis sub formă analogică de către traductoare corespunzătoare. Aceste semnale pot fi atât în curent cât şi în tensiune, de nivel coborât sau ridicat.

Între funcţiile specifice *SADA* pot fi enumerate:

- *selectarea* canalului fizic de acces conform unei adrese primite de la *UCP*;

- eventuale *prelucrări primare* ale informaţiei preluate (liniarizări, corecţii, filtrări, etc);

- *transformarea* semnalelor în vederea compatibilizării cu domeniul dispozitivului (dispozitivelor) de conversie analog-numerică;

- *conversia* din formă analogică în formă numerică;

- *transferul* informaţiei numerice rezultate, în memoria echipamentului numeric prin intermediul magistralei de date.

Corespunzător funcţiilor enumerate, în structura *SADA* pot intra, în totalitate sau nu, următoarele componente:

- *elemente de joncţiune (EJ),* care conectează *SADA* la liniile prin care se transmit semnale de la perifericele de proces;

- *multiplexoare (MUX),* care creează posibilitatea utilizării în comun a resurselor unice;

- *elemente de prelucrare primară(EPP),* care asigură prelucrări de tipul celor enumerate mai sus;

- *amplificatoare*, care permit aducerea semnalului în domeniul convertorului analog-numeric şi adaptarea de impedanţă;

- *dispozitive de eşantionare şi reţinere (DER),* care reprezintă memorii analogice destinate păstrării semnalului eşantionat pe durata conversiei analog - numerice;

- *convertoare analog - numerice (CAN),* destinate conversiei semnalelor preluate din formă analogică în formă numerică;

- *registre tampon ale mărimii convertite (RT)* , care memorează valorile numerice ale mărimilor achiziţionate, până când acestea sunt preluate pe magistrala de date;

- *blocul de comandă(BC)* destinat secvenţierii operaţiilor aferente achiziţiei şi conversiei, ţinând cont de caracterul asincron al acestora în raport cu funcţionarea *UCP*.

În figura 1 sunt prezentate structuri posibile de SADA diferenţiate după numărul de *CAN* utilizate.

Varianta din figura 2 a este caracterizată de prezenţa unui amplificator cu factor de amplificare programabil *(AFAP),* căruia îi succede un lanţ unic *DER-CAN*. Rolul *AFAP* este acela de a aduce semnalele preluate de la traductoare în domeniul de lucru al *CAN*.

Operaţia de achiziţie a datelor de pe un anumit canal, în cazul acestei variantei, se derulează etapizat după cum urmează:

- se transmite multiplexorului analogic *MUXA* adresa canalului selectat;

- se transmite către *AFAP* factorul de amplificare aferent;

- se transmite către *DER* comanda de eşantionare;

- se transmite către *CAN* comanda *Start conversie*;

- după recepţionarea semnalului *Sfârşit conversie* se transmite comanda de înscriere a informaţiei numerice furnizate de *CAN* în registrul tampon *RT*.

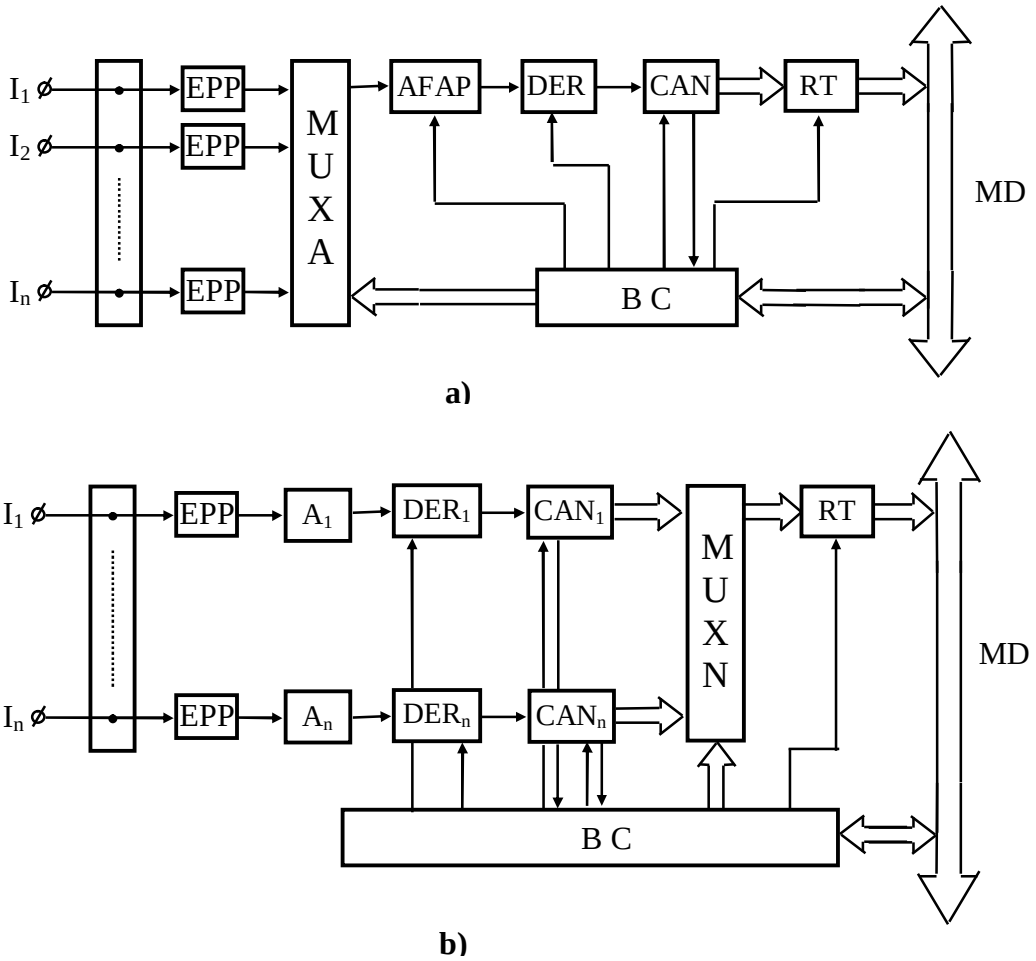

Fig. 2. Structuri de *SADA*:
a - cu un singur CAN şi multiplexare analogică; b - cu mai multe CAN şi multiplexare numerică.

Variantei din figura 2 b îi este specific câte un lanţ *A-DER-CAN* pentru fiecare canal care se conectează la *SADA*. Spre deosebire de varianta anterioară, fiecare din coeficienţii de amplificare ai amplificatoarelor A_1,…, *An* este fix, iar multiplexarea este numerică, resursa comună fiind reprezentată de *RT* iar utilizatorii de ansamblurile *EP-A-DER-CAN*, aferente fiecărui canal.

Achiziţia de pe un canal, în cazul variantei *b* se realizează prin parcurgerea următoarei secvenţe de operaţii:

- se poziţionează multiplexorul numeric *MUXN* pe canalul de pe care urmează a se face achiziţia;

- se transmite comanda de eşantionare către *DER* de pe canalul respectiv;

- se transmite comanda *Start conversie* către *CAN* de pe canalul selectat;

- după recepţionarea semnalului *Sfârşit conversie* se dă comanda de înscriere a codului numeric rezultat în *RT*.

În cazul ambelor variante după înscrierea în registrul tampon *RT*, datele în formă numerică pot fi preluate în memorie prin intermediul magistralei de date *MD*.

Cu toate că în soluţiile prezentate în figura 1 *SADA* se conectează numai la MD, nu este exclusă conectarea în anumite situaţii şi la magistralele de *adrese* şi de *comenzi* ale sistemului respectiv. În continuare se vor face referiri la elementele care intră în structura *SADA*.

3. Subsistemul de achiziţie a datelor numerice

Acest subsistem *(SADN)* asigură accesul pe magistrala de date a sistemului a informaţiei numerice referitoare în mod deosebit la starea procesului sau a unor diviziuni ale acestuia. Dacă se utilizează traductoare cu ieşirea numerică, *SADN* asigură preluarea informaţiei numerice furnizate de acestea.

Un canal multiplu de achiziţie a datelor numerice îndeplineşte următoarele funcţii: - *colectarea* şi centralizarea informaţiei din proces;

- *selectarea* liniilor de intrare şi adaptarea la caracteristicile receptorului;

- *transferul* în memoria sistemului prin intermediul magistralei de date.

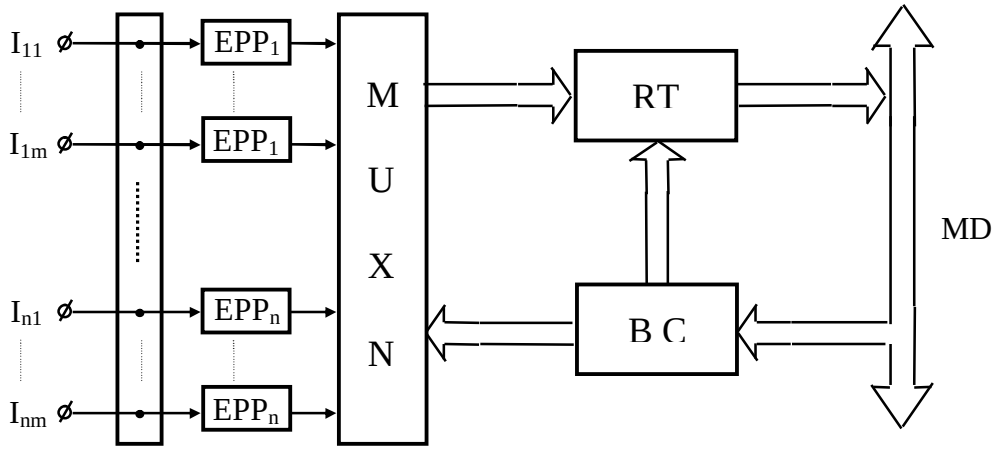

Fig. 3. Structura principialăa unui SADN.

Datele pot fi preluate individual sau pe grupe de 8 sau 16 semnale funcţie de lungimea cuvântului de date a sistemului de calcul. Semnalele pot fi pur *numerice* (niveluri de tensiune, curent sau stări contacte) sau *quasinumerice* (durata sau frecvenţa unor impulsuri). Intrările numerice pot fi *statice*, când sunt

sesizate *nivelele* sau *dinamice,* atunci când sunt sesizate *tranziţiile*. Funcţiile prezentate ale *SADN* pot fi realizate de către structura din figura 3.

Această structură permite achiziţia a *n* grupe fiecare formatădin câte *m* parametri numerici. Elementele de joncţiune asigură cuplarea fizică a liniilor numerice la *SADN*. Elementele de prelucrare primară *EPP* asigură adaptarea semnalelor numerice la cerinţele multiplexorului numeric *MUXN*.

4. Subsistemul de distribuţie a comenzilor analogice (SDCA)

O proporţie însemnată din totalul elementelor de execuţie aferente unui proces tehnologic este reprezentată de cele analogice. *SDCA* asigură conversia semnalelor numerice de comandă în formă analogică şi transmiterea acestora către elementele de execuţie specificate.

Între funcţiile specifice unui *SDCA,* reprezentative sunt următoarele:

- transferul informaţiei numerice de pe magistrala de date a sistemului;
- conversia din formă numerică în formă analogică;
- selectarea canalului aferent elementului de execuţie receptor;
- memorarea comenzii în formă analogică între două momente ale elaborării acesteia.

Corespunzător funcţiilor enumerate, în structura unui *SDCA* pot intra (în totalitate sau nu) următoarele elemente:

- *registre tampon (RT)*, care asigură memorarea comenzilor în formă numerică, din momentul în care acestea sunt disponibile pe magistrala de date şi până la încheierea conversiei numeric-analogice;
- *demultiplexoare (DMUX),* care permit distribuirea unei resurse unice la mai mulţi utilizatori;
- *convertoare numeric-analogice (CNA),* destinate conversiei semnalelor de comandă în formă analogică;
- *memorii analogice (MA),* care memorează comanda în formă analogică între două momente ale elaborării acesteia;
- *bloc de comandă(BC)*, destinat secvenţierii operaţiilor de conversie şi distribuirii comenzilor către proces;
- *elemente de joncţiune (EJ)* , care conectează SDCA la liniile prin care se transmit semnalele de comandă la elementele de execuţie analogice.

În figura 4 sunt prezentate structuri posibile de SDCA diferenţiate după numărul de *CNA.*

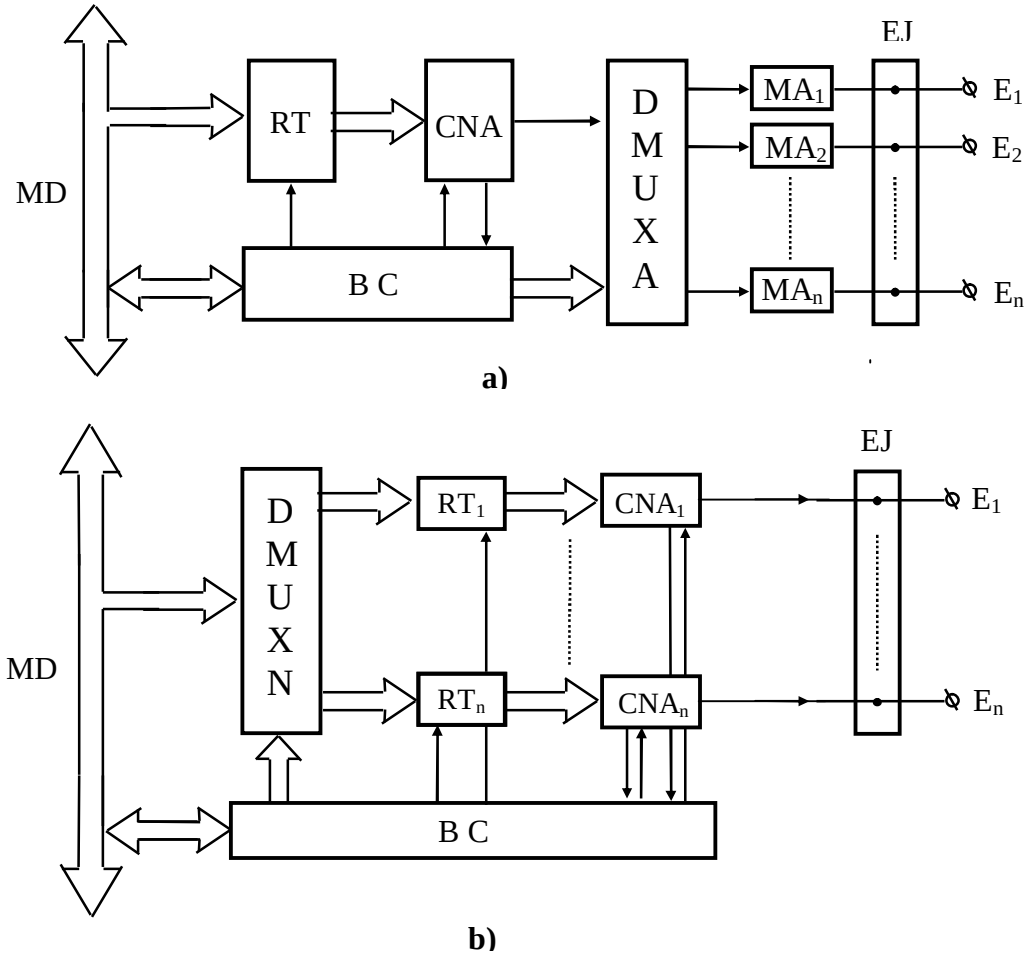

Fig. 4. Structuri posibile de SDCA:
a - cu un singur CNA şi demultiplexare analogică;
b - cu mai multe CNA şi demultiplexare numerică.

Varianta a este caracterizată de existenţa a unui singur *CNA* în calitate de resursă comună a *SDCA*. Operaţia de generare a comenzii pe un anumit canal se desfăşoară conform următoarei secvenţe:

- pe *MD* se transmite valoarea numerică a comenzii şi adresa elementului de execuţie căreia îi este destinată;
- *BC* comandă înscrierea acestei comenzi în *RT*;
- *BC* comandă declanşarea conversiei numeric-analogice;
- la sfârşitul conversiei se poziţionează *DMUXA* pe adresa destinatarului;

- comanda în formă analogică este înscrisă în *MA* aferentă canalului selectat. Din momentul înscrierii în *MA* comanda este accesibilă (prin intermediul *EJ*) elementului de execuţie căruia îi este destinată.

Varianta b este caracterizată de prezenţa câte a unui *CNA* pe fiecare canal. Resursa comună este reprezentată de magistrala de date a sistemului, distribuţia acesteia la canalele utilizator fiind realizată de către demultiplexorul numeric *DMUXN*. Această variantă nu necesită MA deoarece *CNA* de pe fiecare canal asigură memorarea comenzii între două momente ale elaborării acesteia.

Generarea comenzii pe un canal în acest caz se derulează conform secvenţei următoare:

- pe magistrala *MD* se transmite adresa canalului şi valoarea comenzii;
- *BC* poziţionează *DMUXN* pe canalul selectat şi dă comanda pentru declanşarea conversiei;
- la încheierea conversiei comanda este disponibilă pentru elementul de execuţie aferent canalului selectat.

Între cele două variante este de preferat *varianta b,* datorită costurilor ridicate ale memoriilor analogice.

5. Sistemul de distribuţie a comenzilor numerice

Acesta realizează conversia semnalelor primite de la calculator în comenzi numerice cu caracteristici - *nivel, durată, putere*- specifice elementelor cărora le sunt destinate.

Cel mai simplu canal de ieşire este cel de tip releu care asigură comenzi binare de tip închis-deschis, diferenţiate funcţie de tipul releului. În figura 5 este prezentată structura unui *SDCN* care comandă bobina unui releu şi o electrovalvă.

Comenzile se transmit de regulă sub forma unui cuvânt binar, fiecărui element comandat revenindu-i câte un bit. Canalele de ieşire, sub forma unui tren de impulsuri, realizează transformarea valorii numerice primite pe *MD* într-un tren de impulsuri modulate în frecvenţă sau amplitudine. Una din principalele utilizări ale ieşirilor sub formă de impulsuri o reprezintă acţionarea motoarelor pas cu pas *(MPP). MPP* pot fi implicate în comanda referinţelor analogice ale regulatoarelor analogice sau a elementelor de execuţie analogice fără utilizarea *CNA.* De asemenea *SDCN* cu canale de ieşire sub forma unui tren de impulsuri pot comanda elementele de execuţie cu servomotor de tip *MPP*.

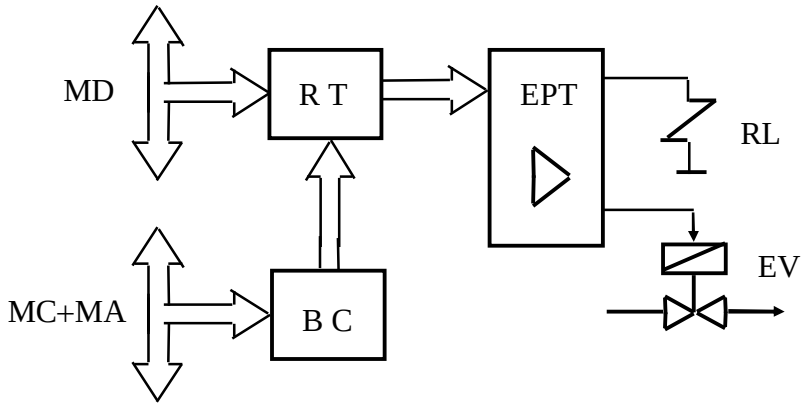

Fig. 5. Structura unui SDCN cu ieşirea pe canal de tip releu: RT - registru tampon; BC - bloc de comandă; EPT - element de prelucrare şi transmitere a semnalelor.

In figura 6 este reprezentată structura principială a unui *SDCN* pentru comanda unui *MPP* care modifică referinţa externă a unui regulator analogic. *BC* primeşte de la *UCP* informaţia referitoare la valoarea referinţei, sensul de variaţie a acesteia şi adresa regulatorului. Referinţa se modifică cu ajutorul *MPP* care în funcţie de numărul impulsurilor aplicate deplasează cursorul potenţiometrului. Se observă că în afara mărimii şi adresei este obligatorie transmiterea sensului de acţionare a *MPP.*

În situaţia în care CNA intră în structura *EE, SDCN* serveşte la interfaţarea echipamentului de conducere cu EE considerat numeric.

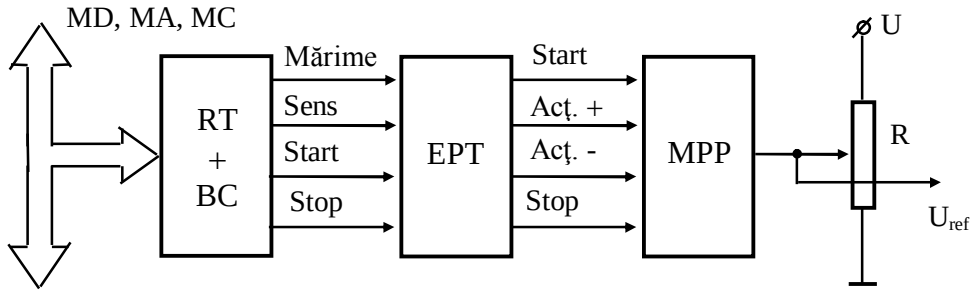

Fig. 6. Structura principialăa unui SADN pentru comanda unui motor pas cu pas: RT + BC - registru tampon şi bloc de comandă; EPT - element de prelucrare şi transmitere; MPP - motor pas cu pas.

Stările şi tranziţiile taskurilor

1. Generalităţi

Aplicaţiile de timp real se dezvoltă apelând la *programarea paralelă sau concurentă.* Două taskuri se numesc *paralele sau concurente*[1] dacă prima instrucţiune a unuia trebuie executată înainte de încheierea ultimei instrucţiuni a celuilalt. O execuţie pur paralelă este posibilă în situaţia în care maşina pe care se execută aplicaţia conţine mai mult de o *Unitate Centrală de Procesare (UCP).* În cazul sistemelor cu o singură *UCP* execuţia este *pseudoparalelă* , timpul acesteia fiind partajat între mai multe taskuri.

Procesarea pseudoparalelă se realizează uzual prin execuţia intercalată a taskurilor[2]. Acest tip de procesare presupune lucrul în întreruperi, taskurile executându-se intercalat cu rutina te tratare a întreruperilor, aspect evidenţiat în figura 2.4.

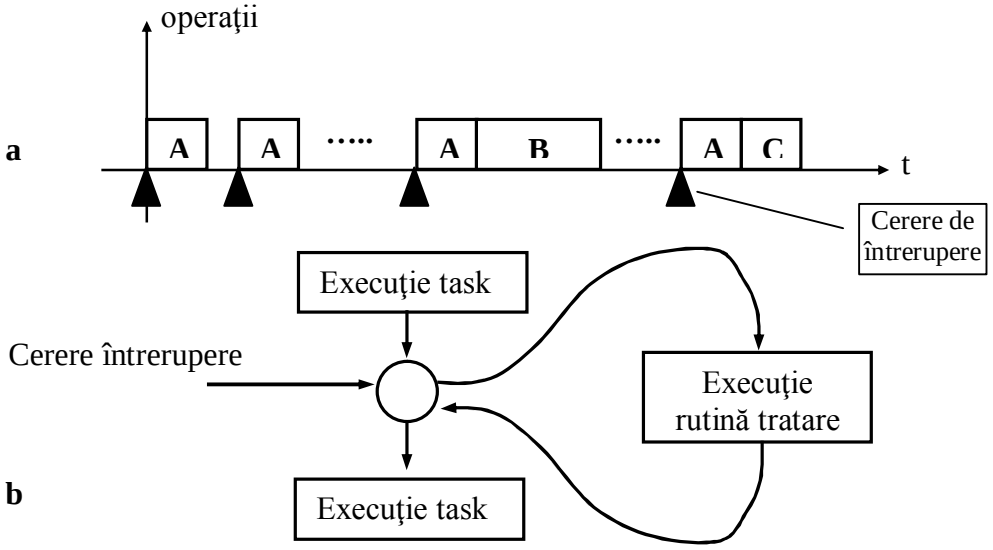

Fig. 1. Execuţia intercalată a taskurilor.

În exemplul din figura 1a se consideră următoarea semnificaţie a taskurilor:

- A – task de achiziţie date de la CAN (iniţiat de un semnal de întrerupere de la CAN la încheierea conbersiei);
- B – task de calcul comenzi (se executa la anumite intervale de timp);

[1] Atributul de paralel are în vedere execuţia *paralelă sau pseudoparalelă* a mai multor acţiuni. Atributul *concurent* se referă la faptul că taskurile se află în competiţie pentru deţinerea de resurse.

[2] În mod obişnuit un *task* reprezintă o unitate elementară de program independentă din punct de vedere logic.

- C – task de avertizare (se execută la neîncadrarea între limite a mărimii achiziţionate din proces).

În figura 1b este reprezentată execuţia rutinei de tratare a unei întreruperi. Se remarcă faptul că execuţia acestei rutine presupune întreruperea şi apoi reluarea unui task.

În afara lucrului în întreruperi , execuţia intercalată a taskurilor mai presupune existenţa în cadrul sistemului de operare a mecanismelor pentru:

- salvarea şi restaurarea stării unui task înainte, respectiv după întrerupere;
- utilizarea neconflictuală a resurselor critice *(excludere mutuală);*
- coordonarea execuţiei taskurilor interactive *(sincronizare);*
- transferul de informaţie între taskuri *(comunicare);*
- activarea şi dezactivarea taskurilor.

Cele cinci mecanisme sunt implementate în *nucleul Sistemului de Operare în Timp Real Multitasking (SOTRM),* nucleu care mai este cunoscut sub denumirea de *executiv*. Acesta se află în vecinătarea imediată secţiunii hardware şi care se consideră că face parte din aplicaţia de timp real. După cum se observă din figura 2, o asemenea aplicaţie mai conţine taskuri sistem şi taskuri utilizator.

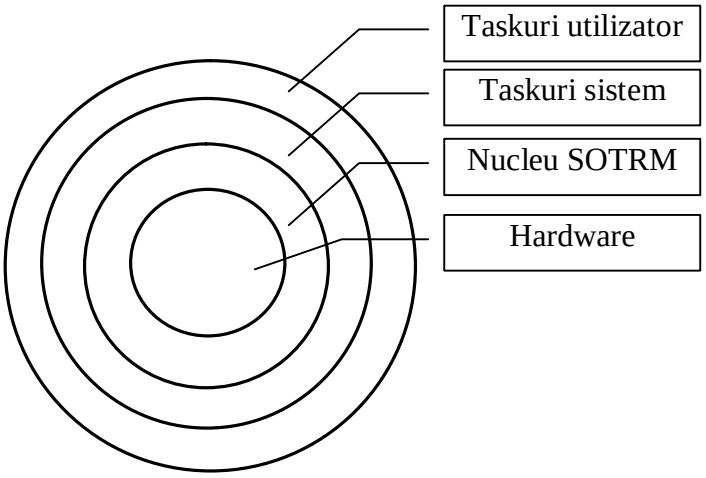

Fig. 2. Structura stratificată a unei aplicaţii de timp real.

O asemenea execuţie implică existenţa în cadrul *SOTRM* a mecanismelor pentru:

- salvarea şi restaurarea stării taskurilor;

- utilizarea neconflictuală a resurselor critice *(excluderea mutuală);*
- coordonarea taskurilor interactive *(sincronizarea);*
- reglementarea schimbului de date între taskuri *(comunicarea);*
- activarea şi dezactivarea taskurilor.

În mod obişnuit un task are asociate în memoria internă trei secţiuni şi anume:

- secţiunea de date – conţine operanzi şi rezultate ;
- secţiunea de cod – conţine codul care implementează algoritmul aferent aplicaţiei realizate de task;
- secţiunea de stivă – în care sunt salvate informaţii care să permită refacerea contextului la reluarea execuţiei taskului după o întrerupere.

În structura unei aplicaţii de timp real sunt incluse în mod obişnuit *taskuri interactive şi taskuri disjuncte.*

- *Taskurile interactive* sunt taskurile care pe parcursul evoluţiei lor utilizează resurse în comun şi fac schimb de informaţie .
- *Taskurile disjuncte* sunt taskuri care nu interacţionează, respectiv care nu utilizează resurse în comun şi nu fac schimb de informaţie.

Evoluţia unei aplicaţii care conţine numai taskuri disjuncte este unică indiferent de viteza şi ordinea de execuţie a taskurilor.

În ceea ce priveşte evoluţia unui sistem de taskuri interactive , aceasta este influenţată de doi factori şi anume:

- modul de planificare a execuţiei taskurilor;
- modul de efectuare a tranziţie între stări şi/sau substări.

Execuţia unei aplicaţii de timp real presupune evoluţia taskurilor într-un spaţiu al stărilor, în care tranziţiile sunt provocate de directive care pot fi emise de către planificator sau de către taskul aflat în execuţie. Există mai multe abordări referitoare la stările şi tranziţiile taskurilor, în cele ce urmează fiinde prezentată o naastfel de abordare.

2. Stările taskurilor

În această abordare un task se poate găsi într-una dintre următoarele stări:

- neinstalat;
- inactiv;
- activ,

stării activ fiindu-i asociate una dintre substările:

- execuţie;
- gata de execuţie
- blocat.

Aceste stări împreună tranziţiile posibile sunt ilustrate în graful din figura 3.

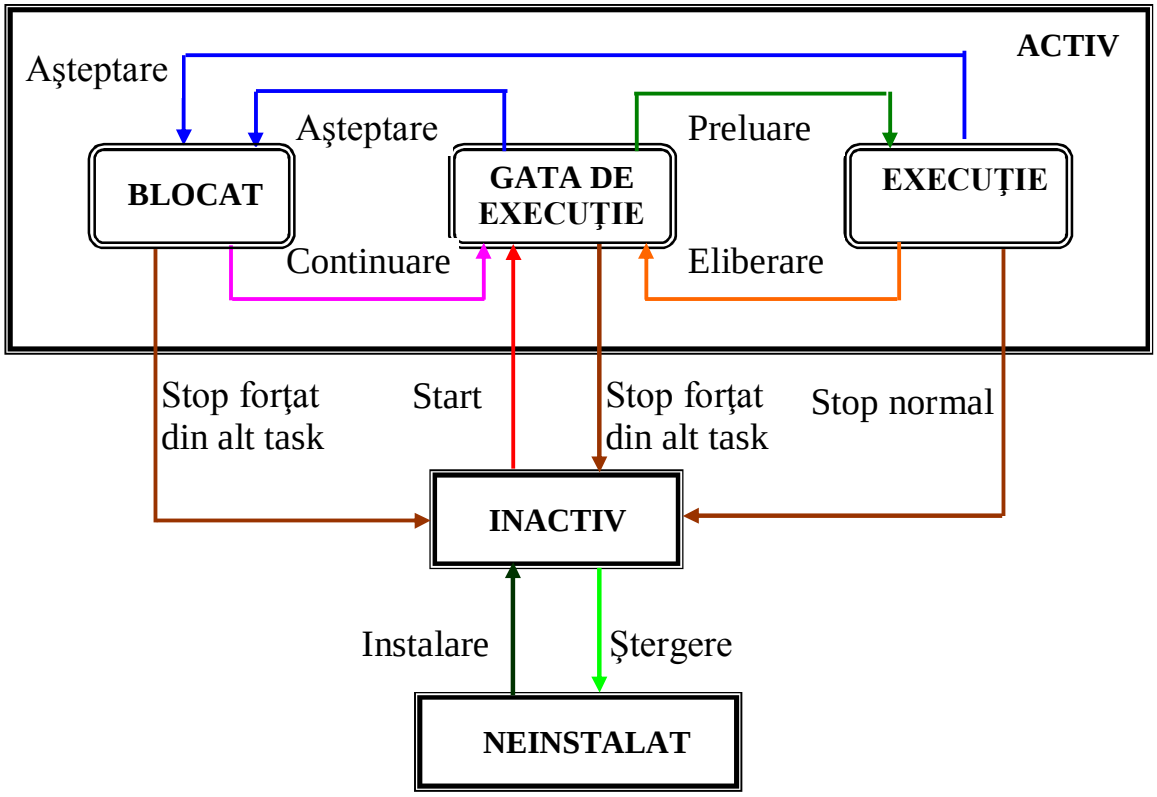

Fig. 3. Evoluţia taskurilor în spaţiul stărilor stărilor şi substărilor.

- Un task *neinstalat* este un task rezident în memoria internă sau externă, necunoscut de executivul de timp real.
- Un task *inactiv* este un task instalat, pentru care fie nu s-a făcut un apel de rulare fie şi-a încheiat rularea (prin stop natural sau forţat).
- Un task *activ* este un task rulabil care se poate găsi într-una din substările *execuţie, gata de execuţie, blocat.*
- Substarea *execuţie* corespunde situaţiei în care *taskul* deţine controlul *UCP.* La sistemele cu o singură unitate centrală, un singur task se poate găsi în această substare.

- Substarea *gata de execuţie* este specifică taskurilor care aşteaptă să preia controlul *UCP.* Indicatorii asociaţi acestora se înscriu într-o coadă, execuţia putându-se efectua în două moduri:

- ciclic *(ROUND – ROBIN)* , taskurile fiind executate succesiv în interiorul unei cuante de timp sau până la terminarea normală;
- după priorităţi, controlul *UCP* fiind preluat de taskul cu prioritatea cea mai ridicată.

- Substarea *blocat* este asociată taskurilor care se pot găsi într-una din situaţiile:

- aşteaptă să i se aloce memorie;
- aşteaptă producerea unui eveniment;
- aşteaptă realizarea unei condiţii de timp;
- aşteaptă realizarea unei operaţii de intrare / ieşre.

3. Tranziţiile taskurilor

Tranziţiile între stări şi substări se pot realiza în două moduri şi anume:

- prin întreruperi hardware sau software, contextul comutării fiind determinat de nivelul întreruperii;
- prin apeluri către *EXECUTIV* din taskul aflat în *EXECUŢIE.* Acestea sunt apeluri de subrutine neîntreruptibile numite *PRIMITIVE sau DIRECTIVE,* contextul comutării fiind determinat de numele şi parametrii directivei.

Există două categorii de *directive* şi anume:

- directive care *nu declară un eveniment semnificativ* , după a căror execuţie controlul *UCP* este returnat taskului întrerupt;
- directive care *declară un eveniment semnificativ* , după a căror execuţie controlul *UCP* este returnat taskului aflat pe prima poziţie în coada taskurilor *GATA DE EXECUŢIE.*

În cele ce urmează se prezintă tranziţiile ilustrate în graful din figura 3 şi directivele asociate.

- Tranziţia *NEINSTALAT – INACTIV* se realizează prin directiva ***Instalare*** care face cunoscut executivului taskul realizând în acelaşi timp crearea şi alocarea *blocului descriptor al taskului (BDT)* care poate conţine următoarele elemente:

- parametrii ai taskului (*adresă de început, prioritate, identificator, nume, etc.*);
- mărimea stivei;
- indicatori către alte taskuri.

- Tranziţia *INACTIV –NEINSTALAT* se realizează prin directiva ***Ştergere*** care elimină taskul din lista taskurilor cunoscute de către executiv şi dezactivează *BDT.*

- Tranziţia *INACTIV – ACTIV* care presupune trecerea taskului în substarea *GATA DE EXECUŢIE* se realizează prin directiva ***Start.*** Această directivă semnifică practic un apel de rulare pentru respectivul task.

- Tranziţia *GATA DE EXECUŢIE – EXECUŢIE* se realizează printr-o directivă de ***Preluare.*** După execuţia acestei directive controlul este cedat taskului din prima poziţie a listei de aşteptare la procesor. După cum se va arăta ulterior această listă poate fi organizată după priorităţi sau respectând principiul rotaţiei.

- Tranziţia *GATA DE EXECUŢIE – BLOCAT* se realizează printr-o directivă de ***Aşteptare,*** în situaţia în care taskului nu i se mai poate aloca memorie pentru stivă dacă ar prelua controlul *UCP.*

- Tranziţia *GATA DE EXECUŢIE – INACTIV* se realizează printr-o directivă de terminare forţată ***(Stop forţat)*** directivă lansată de către taskul aflat în execuţie. Această tranziţie presupune menţinerea *BDT* pentru respectivul task.

- Tranziţia *EXECUŢIE – GATA DE EXECUŢIE* se realizează printr-o directivă de ***Eliberare***. Controlul *UCP* este cedat ca urmare a declarării unui eveniment ca fiind semnificativ (cum ar fi expirarea cuantei de timp alocate, sau trecerea în starea *GATA DE EXECUŢIE* **,** a unui task cu prioritate superioară**.**

- Tranziţia *EXECUŢIE – BLOCAT* se realizează printr-o directivă de ***Aşteptare,*** în situaţia în care continuarea execuţiei taskului este condiţionată de: *producerea unui eveniment extern, realizarea unei condiţii de timp, realizarea unei operaţii de intrare-ieşire, etc.*

- Tranziţia *EXECUŢIE – INACTIV* care presupune păstrarea *BDT* se realizează la o terminare normală a taskului ***(Stop normal)***.

- Tranziţia *BLOCAT – GATA DE EXECUŢIE* se realizează printr-o directivă de ***Continuare ,*** lansată de către executiv dacă situaţia care a determinat blocarea a dispărut.

- Tranziţia *BLOCAT – INACTIV* se realizează printr-o directivă de terminare forţată ***(Stop forţat)*** directivă lansată de către taskul aflat în execuţie. Ca

şi în cadrul altor tranziţii în starea *INACTIV* şi aceasta presupune menţinerea *BDT* pentru respectivul task.

Este de menţionat faptul că tranziţiile taskurilor în spaţiul stărilor trebuie tratate în strânsă corelaţie cu operaţiile multitasking aferente rezolvării problemei interacţiunii dintre taskuri, operaţii care vor fi tratate pe parcursul acestui capitol.