

POLITEHNICA University of Timisoara
Department of Computer Science & Engineering
Digital Signal Processing Laboratories – DSPLabs
2, V. Parvan Bv., 1900 – Timisoara, ROMANIA
Tel: +40 256 403271
Web: <http://dsplabs.utt.ro>



Mihai V. MICEA

**Proiectarea și implementarea
sistemelor timp-real stricte
pentru aplicații critice.
Studiu de caz: sistemul HARETICK**

Referat de doctorat #3

**Conducător științific
Prof. dr. ing. Vladimir CREȚU**

2003

Cuprins

1	Introducere	1
2	Nucleul de operare HARETICK: caracteristici generale	3
2.1	Modelul de sistem TR strict pentru aplicații critice	3
2.2	Caracteristicile generale ale nucleului HARETICK.....	6
3	Componentele HARETICK și relația dintre acestea	9
3.1	Componentele HARETICK.....	9
3.1.1	Încărcarea și lansarea sistemului: BOOT	11
3.1.2	Inițializarea sistemului: SYSINIT	11
3.1.3	Comunicația de date: DATALINK	11
3.1.4	Interfațarea cu operatorul: MONITOR	12
3.1.5	Încărcarea unei aplicații: LOADER	12
3.1.6	Planificatorul task-urilor TR lejere: SSCD	13
3.1.7	Planificatorul ModX-urilor: HSCD.....	13
3.1.8	Comutarea execuției ModX-urilor: HDIS.....	14
3.1.9	Raportarea stării curente a sistemului: STATREPO	15
3.1.10	Subrutina de raportare a evenimentelor sistem: TiLT.....	16
3.2	Relația dintre componentele nucleului HARETICK	16
4	Reprezentarea și gestionarea timpului	19
4.1	Timpul ca parametru esențial de operare al sistemului	19
4.2	Structurile destinate reprezentării și gestionării timpului în HARETICK	21
4.2.1	Timpul absolut sistem	21
4.2.2	Ceasul timp-real	22
4.2.3	Timpul de planificare	24
4.3	Mecanismele de sincronizare a structurilor de gestionare a timpului în HARETICK.....	25
5	Reprezentarea ModX-urilor în cadrul sistemului	27
5.1	Modelul task-ului TR strict: ModX-ul.....	27
5.2	Reprezentarea ModX-urilor în cadrul HARETICK	30
5.2.1	Reprezentarea ModX-urilor în memorie	31
5.2.2	Tabela de simboluri.....	33
5.2.3	Tabela descriptorilor de procese	33
5.3	Operarea de principiu a ModX-urilor în cadrul HARETICK.....	35
5.4	Concluzii.....	39
6	Funcționarea de principiu a sistemului	41
6.1	Contextele de execuție sistem și comutarea acestora	41
6.2	Stările ModX-urilor	47

7 Executivul TR al HARETICK: HDIS	53
7.1 Caracteristici generale.....	53
7.2 Structurile de date utilizate de HDIS: Tabela de Dispatch	55
7.3 Funcționarea de principiu.....	57
8 Concluzii și perspective	63
8.1 Concluzii	63
8.2 Stadiul actual al implementării și perspective.....	67
Bibliografie	69

1 Introducere

Referatul de față face parte din programul de doctorat cu tema "Contribuții la achiziția și prelucrarea numerică în timp-real a semnalelor utilizând sisteme de calcul distribuite", sub coordonarea științifică a prof. dr. ing. Vladimir CREȚU.

Sistemele de control digital încorporat (embedded systems) și achiziția și prelucrarea numerică a semnalelor (DSP-based systems) sunt două domenii înrudite, de interes major în preocupările actuale ale comunității academice și industriale, fapt dovedit și de numărul imens de aplicații ce integrează astfel de sisteme în aproape toate domeniile activității umane. O cerință esențială impusă acestor sisteme și aplicații este operarea în timp-real, cu garantarea respectării termenelor de timp impuse de specificațiile de proiectare și de mediu. Un număr foarte mare de aplicații au un impact critic asupra mediului înconjurător și/sau asupra factorului uman cu care interacționează. Eșecul unor astfel de sisteme în îndeplinirea *la termenele specificate* a sarcinilor programate poate avea consecințe catastrofice asupra mediului înconjurător sau ducând la pierderi de vieți omenești.

În cadrul lucrării a fost abordată problematica sistemelor timp-real stricte destinate aplicațiilor critice de achiziție și prelucrare numerică a semnalelor (DSP-based systems) și de control digital încorporat (embedded systems), din perspectiva proiectării și implementării acestora.

Lucrarea de față este o continuare a preocupărilor prezentate în referatul al doilea, în cadrul căruia a fost conceput și analizat un model complet, unitar, de sistem TR strict pentru aplicații critice, OPEN-HARTS.

Referatul descrie principiile și metodele implicate în proiectarea și implementarea unui nucleu de operare TR strict, detaliind particularitățile întâlnite cu ajutorul unui studiu de caz: implementarea nucleului de operare HARETICK pe arhitectura Motorola DSP56307.

Capitolul 2 prezintă nucleul HARETICK ca parte componentă a sistemului OPEN-HARTS, alături de mediul de dezvoltare și analiză a aplicațiilor TR, INVERTA. Sunt trecute în revistă caracteristicile generale ale HARETICK, caracteristici care derivă din faptul că au fost vizate sistemele de control digital încorporat (embedded systems).

HARETICK este conceput ca un set omogen de componente software de bază, care interacționează strâns unul cu celălalt, asigurând ca ansamblu, operarea platformei țintă (Motorola DSP56307 Evaluation Module) conform principiilor TR vizate. Descrierea componentelor HARETICK (tipul, rolul, principalele operații pe care le implementează), precum și a relațiilor dintre ele, este realizată în capitolul următor.

Timul, ca parametru esențial de operare al sistemului, este reprezentat și gestionat cu ajutorul unor structuri și mecanisme dedicate. Capitolul 4 prezintă elementele implicate în implementarea celor trei structuri de gestiune a timpului în HARETICK: timpul sistem absolut, ceasul de timp-real și timpul de planificare. Sunt

descrise de asemenea, relațiile stabilite între aceste structuri și mecanismele de sincronizare aferente.

Task-urile TR stricte sunt modelate, în concepția lucrării de față, prin ModX-uri – task-uri periodice, modulare, cu specificații complete și stricte de comportare în timp și cu planificare și execuție în context non-preemptiv. Un ModX conține seturi de parametri de intrare și de ieșire, variabile globale, un set de parametri ce îi specifică comportarea în timp, seturile semnalelor de intrare și de ieșire cu care interacționează, și secvența de cod program (specificarea funcțională). În Capitolul 5 sunt detaliate structurile de date concepute pentru reprezentarea ModX-urilor în cadrul nucleului HARETICK și utilizarea acestor elemente pe parcursul existenței și operării ModX-urilor în sistem.

Funcționarea de principiu a sistemului se bazează pe cele două contexte de execuție complementare – contextul strict și cel lejer, și pe comutarea acestora, realizată de către executivul TR al HARETICK, HDIS.

Contextul strict de execuție poate fi privit ca o "întrerupere prelungită" a celui lejer, conform unei planificări riguroase, stricte, analizată apriori din punctul de vedere al fezabilității (analiză "offline").

Un alt mod de a vedea problema este acela că, deasupra contextului lejer (reprezentând modul de operare tradițional, preemptiv, al sistemelor de calcul) se poate implementa, ca o facilitate de execuție cu garantarea respectării termenelor stricte și cu predictibilitate maximă, contextul strict, a cărui întrerupere (RTC) este cea mai prioritară în sistem. În plus, pe durata execuției în contextul strict, celelalte întreruperi sunt suspendate.

Capitolul 6 detaliază funcționarea sistemului HARETICK, insistând asupra celor două contexte de execuție și a modului de comutare a acestora. Sunt prezentate de asemenea stările ModX-urilor și tranzițiile posibile între stări.

În capitolul următor se face o prezentare a componentei principale a nucleului HARETICK – executivul timp-real HDIS. Sunt detaliate structurile de date utilizate și organigramele de funcționare ale HDIS.

Concluziile desprinse pe parcursul lucrării, o prezentare a stadiului actual al implementării nucleului HARETICK, precum și perspectivele de cercetare și dezvoltare ulterioară, încheie referatul.

2 Nucleul de operare HARETICK: caracteristici generale

2.1 Modelul de sistem TR strict pentru aplicații critice

Unul din rezultatele abordării problematicei sistemelor TR stricte pentru aplicații critice de prelucrare numerică a semnalelor și control digital încorporat din cadrul activității noastre de doctorat, este conceperea unui model pentru astfel de sisteme, descris în referatul anterior.

Modelul propus se bazează pe câteva considerente principale care definesc liniile generale urmate în proiectarea și dezvoltarea acestuia:

- Utilizarea întreruperilor în sistemele TR stricte reprezintă o problemă din punctul de vedere al predictibilității, afectând capacitatea acestora de a garanta în orice condiții de operare respectarea termenelor impuse task-urilor TR stricte.
- Pentru a oferi predictibilitate maximă sistemelor TR stricte, este necesară abordarea modelelor non-preemptive în ceea ce privește definirea semnalelor, a task-urilor, conceperea algoritmilor de planificare și implementarea sistemului.
- În cadrul dezvoltării și implementării aplicațiilor TR, faza de analiză offline, apriori execuției pe sistemul țintă, a setului de task-uri TR stricte al aplicației, reprezintă o cerință obligatorie.
- Dezvoltarea viabilă a sistemelor și aplicațiilor TR stricte trebuie să aibă la bază, pentru toate fazele sale, modele și metode omogene, compatibile.

În consecință, modelul de sistem TR conceput cuprinde două elemente principale: un mediu integrat, destinat dezvoltării și analizei offline a aplicațiilor, și un nucleu de operare TR strict pe care se încarcă și se execută aplicațiile validate. OPEN-HARTS (Operanting Environment for Hard Real-Time Systems) este conceput ca un mediu de operare pentru sisteme și aplicații TR stricte, având două componente majore (vezi Figura 2-1):

- (1) Mediul integrat de dezvoltare și analiză a aplicațiilor TR stricte pe sisteme DSP și de control digital încorporat, INVERTA (Integrated Visual Environment for Real-Time Application Analysis and Development). INVERTA oferă programatorului de sistem și de aplicații TR un mediu vizual integrat, destinat proiectării, specificării și verificării formale a aplicațiilor (cu accent pe comportamentul temporal al task-urilor – obligatoriu la cele TR stricte). INVERTA permite de asemenea, programarea funcțională a fiecărui task al aplicației, analiza temporală a task-urilor (extragerea WCET) și analiza planificabilității. Toate operațiile se realizează în regim "off-line" vis-a-vis de

operarea sistemului TR propriu-zis, pe o platformă de calcul (Host platform), care poate fi statică (workstation), sau mobilă (Laptop, PDA, Tablet PC, etc.).

- (2) Nucleul de operare TR hibrid pentru sisteme încorporate, HARETICK (Hard Real-Time Compact Kernel). HARETICK este instalat pe platforma țintă de control digital încorporat, și permite încărcarea și execuția task-urilor TR stricte (Hard Real-Time tasks) în regim non-preemptiv și a celor lejere (Soft Real-Time tasks) în regim preemptiv.

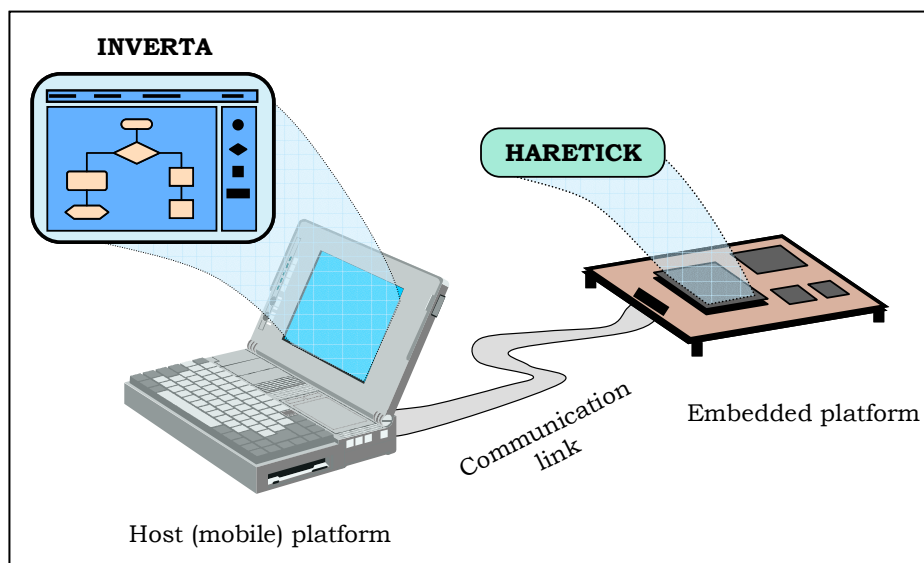


Figura 2-1. Arhitectura generală a sistemului OPEN-HARTS

Cele două componente ale sistemului OPEN-HARTS sunt interconectate printr-o legătură și un protocol de comunicații, permițând schimbul de date, comenzi și informații de stare între INVERTA (programatorul de sistem/aplicații) și HARETICK (sistemul TR care operează pe o platformă țintă).

Întregul sistem se bazează pe o concepție puternic modulară, atât în ceea ce privește arhitectura generală cât și modul de abordare a proiectării, specificării, analizei și execuției aplicațiilor TR.

Mediul integrat INVERTA permite programatorului de sistem/aplicații dezvoltarea unei aplicații TR în regim offline, fără a interfera deci, în această fază, cu sistemul țintă, pe care poate opera la un moment dat o versiune anterioară a aplicației. Dezvoltarea aplicației începe prin construirea într-un mediu vizual și într-o manieră modulară, a diagramei de task-uri (ModX-uri și task-uri TR lejere), rezultând o rețea vizuală, echivalentă cu graful dependențelor de program (*PDG – Program Dependence Graph*). În continuare, pentru fiecare nod al grafului (reprezentând de fapt un task – ModX sau task TR lejer), se vor specifica următoarele elemente caracteristice:

- Setul parametrilor de intrare și de ieșire (dependența de date față de celelalte module).
- Setul variabilelor globale ale aplicației, utilizate de către task.

- Setul semnalelor de intrare și de ieșire cu care interacționează task-ul. Pentru fiecare semnal, programatorul poate modela comportarea acestuia în timp, prin intermediul unui set minimal și complet de parametri temporali. Pentru semnalele de intrare, specificarea comportării temporale este obligatorie.
- Prin intermediul unui set minimal și complet de parametri temporali, programatorul poate modela comportarea în timp a task-urilor TR stricte – ModX-urile.
- Comportarea funcțională a fiecărui task se va specifica prin intermediul unui limbaj de programare specific aplicațiilor TR (limbaj de asamblare, C modificat, etc.).

După proiectarea și specificarea aplicației, INVERTA va trece în mod automat la faza de verificare/validare a specificațiilor. Sistemul va utiliza seturi de formule bine stabilite, între parametrii temporali ai semnalelor și ai task-urilor aplicației (vezi referatul anterior), atât pentru a verifica validitatea specificațiilor, cât și pentru a stabili în mod automat valorile parametrilor ce nu au fost specificați explicit de programator. INVERTA verifică, de asemenea, dependențele de control și de date ale aplicației, ținând cont și de specificarea funcțională (codul program) al fiecărui task, și modifică acolo unde e cazul, afișarea acestor dependențe în mod corespunzător.

După validarea întregii aplicații, atât din punct de vedere funcțional, cât și al comportamentului în timp, INVERTA va compila codurile sursă ale fiecărui task și va efectua analiza temporală a programelor, pentru obținerea timpilor maximi de execuție pentru fiecare task (*WCET*). Tot în cadrul fazei de analiză a aplicației, INVERTA aplică algoritmi planificare special concepuți pentru modelul de sisteme TR stricte considerate în lucrare, pentru a testa planificabilitatea setului de ModX-uri ale aplicației.

În cazul în care INVERTA a găsit o planificare fezabilă a aplicației, se efectuează pregătirile finale pentru încărcarea ei pe platforma țintă, unde va fi executată sub nucleul de operare HARETICK: link-editare, generarea tabelii de procese (ce conține și parametrii temporali ai task-urilor) și a grafului aplicației, generarea tabelilor de simboluri și a hărții de memorie, etc.

Legătura de date și comenzi (DATALINK) dintre INVERTA și HARETICK are următoarele caracteristici principale:

- Implementează un protocol de comunicații simplu, dar flexibil și scalabil.
- Permite transmiterea de comenzi utilizator prin intermediul INVERTA, către nucleul de operare HARETICK, care execută o aplicație TR oarecare.
- Permite citirea stării de operare a sistemului țintă, prin comandarea generării de diverse rapoarte de execuție.
- Permite încărcarea unei aplicații noi (sau modificate) în sistemul țintă, sau descărcarea informațiilor complete despre o aplicație ce se execută pe sistemul țintă. Datele ce caracterizează execuția unei aplicații TR oarecare pe platforme HARETICK, cuprind printre altele, următoarele elemente:
 - sursele de cod compilat ale task-urilor aplicației;
 - tabela descriptorilor de procese (cu parametrii temporali ai task-urilor);
 - tabela ce descrie graful aplicației;
 - harta de memorie;
 - tabela de simboluri.

Așadar, prin intermediul legăturii de date, a protocolului de comunicație și a interfețelor corespunzătoare din INVERTA și HARETICK, programatorul de aplicații poate să citească la orice moment starea sistemului țintă (în timpul operării acestuia, și fără a afecta operarea sistemului) – incluzând informațiile complete despre aplicația executată curent, poate să trimită diverse comenzi utilizator către sistem și să încarce o nouă aplicație pentru a o înlocui pe cea curentă.

La terminarea cu succes a încărcării unei aplicații în cadrul sistemului țintă, nucleul de operare HARETICK va începe execuția acesteia în timp real. Acest moment definește practic începerea operării efective a aplicației în cadrul sistemului, și predictibilitatea execuției, asupra căreia se concentrează activitatea noastră de cercetare, este garantată începând din acest moment.

2.2 Caracteristicile generale ale nucleului HARETICK

Conceperea, proiectarea și implementarea nucleului HARETICK are la bază faptul că sunt vizate sistemele de control digital încorporat (embedded systems). Prin urmare, HARETICK este un nucleu de operare timp-real de tip *single-user*: programatorul de sistem/aplicații poartă întreaga responsabilitate a proiectării, programării și utilizării sistemului.

De asemenea, HARETICK este un nucleu *multitasking*, permițând execuția concurentă, planificată a unor seturi compuse din mai multe task-uri. Nucleul oferă facilități pentru operarea cu predictibilitate maximă a aplicațiilor critice, garantând respectarea termenelor stricte de timp impuse de acestea. În continuare vom prezenta o serie de alte caracteristici și proprietăți importante ce definesc arhitectura și operarea nucleului HARETICK.

Arhitectură modulară

Sistemul HARETICK este conceput și proiectat într-o manieră puternic modularizată, vizându-se simplitatea și reducerea pe cât posibil a dimensiunilor acestuia. Pentru contextul strict de execuție a task-urilor, sistemul nu efectuează nici un fel de analize de planificabilitate și fezabilitate, ci implementează algoritmi de planificare, cu comportare temporală cunoscută, asupra aplicației analizate și optimizate apriori, offline.

Toate componentele software ale sistemului și ale aplicațiilor dezvoltate pentru acesta, sunt proiectate și implementate sub formă de module software independente, care comunică între ele prin intermediul parametrilor de intrare/ieșire și al variabilelor globale.

Task-urile existente la un moment dat în cadrul HARETICK sunt împărțite în două categorii: task-uri sistem, care definesc practic nucleul de operare și sunt încărcate în faza inițială de pornire a sistemului, și task-uri aplicație, care implementează o aplicație oarecare, încărcată la un moment dat în sistem. Nucleul HARETICK permite încărcarea, descărcarea sau înlocuirea unei aplicații TR la orice moment, prin intermediul unor comenzi operator specifice.

Gestionarea timpului

Timpul, ca parametru esențial de operare al sistemului, este reprezentat și gestionat cu ajutorul unor structuri și mecanisme dedicate. *Baza de timp* a sistemului este implementată de *ceasul timp-real* (RTC) – elementul activ de gestionare a timpului în HARETICK. RTC are două roluri principale: (i) contor de timp, utilizat pentru referiri ale componentelor sistemului la instanța curentă de timp, și (ii) singura sursă (predictibilă) de întreruperi. RTC este o sursă predictibilă de întreruperi în sensul că se poate determina oricând, și de către orice task, momentul apariției următoarei întreruperi.

Pe lângă RTC, nucleul HARETICK gestionează încă două structuri de timp. *Timpul absolut sistem*, este utilizat pentru referiri ale operatorului și ale task-urilor din sistem la timpul absolut, și poate fi setat (modificat) de către operatorul sistemului în orice moment, prin intermediul unor comenzi specifice. *Timpul de planificare* este utilizat și gestionat de către planificatorul contextului strict de execuție al sistemului pentru calcularea și specificarea momentelor la care sunt planificate execuțiile task-urilor TR stricte.

Spre deosebire de timpul absolut sistem, celelalte două structuri – ceasul timp-real și timpul de planificare, lucrează practic în domeniu temporal relativ. Pe de altă parte, toate cele trei tipuri de timp cu care lucrează nucleul HARETICK sunt sincronizate între ele.

Gestionarea proceselor

Procesele (task-urile) cu care operează sistemul HARETICK sunt organizate, reprezentate și gestionate în manieră modulară, utilizând structuri de memorie și mecanisme specifice pentru fiecare task în parte: zonă dedicată pentru cod, zonă pentru parametrii de ieșire, zonă pentru variabilele globale, zonă comună pentru parametrii de intrare și variabilele locale, utilizată de către task-ul aflat în execuție curentă, tabela cu graful aplicației, tabela cu identificatorii de procese, tabela de simboluri, etc.

Task-urile cu care operează nucleul HARETICK sunt împărțite în două categorii principale: (1) task-uri TR stricte (task-uri HRT – Hard Real-Time) denumite ModX-uri (Module eXecutabile), și (2) task-uri lejere (task-uri SRT – Soft Real-Time). În consecință, sistemul HARETICK are două contexte de execuție complementare: contextul strict (sau contextul HRT) în care task-urile TR stricte din sistem (ModX-urile) se execută în regim non-preemptiv, și contextul lejer (sau contextul SRT), pentru execuția task-urilor lejere în regim preemptiv.

Din acest punct de vedere, HARETICK este un *nucleu de operare hibrid*, comutarea celor două contexte de execuție fiind realizată de un task sistem special – executivul TR al HARETICK.

Gestionarea memoriei

Principalul deziderat urmărit la proiectarea și implementarea sistemului HARETICK este operarea predictibilă, cu garantarea respectării termenelor stricte de timp pentru task-urile care au aceste cerințe. Prin urmare toate structurile și

mecanismele implementate de nucleu au o arhitectură și o comportare statică, bine determinată și limitată în timp și spațiu.

Memoria este gestionată în mod liniar de către nucleu, și este împărțită în două secțiuni – memoria sistem și memoria aplicație. Cu toate acestea, accesul task-urilor la toate zonele de memorie este permis, nefiind implementat nici un mecanism care să le restricționeze și să definească zone de memorie "protejate". Motivația abordării este dată de caracteristica principală pentru care este proiectat sistemul: nucleu de operare TR pentru sisteme încorporate (embedded systems), în cadrul căruia programatorul/operatorul are întregul control – și responsabilitate.

Mecanismele IPC

Comunicația dintre task-uri (IPC – Inter-Process Communication) este realizată prin intermediul parametrilor de intrare/ieșire și al variabilelor globale. ModX-urile, ca task-uri stricte, periodice, care se execută în context non-preemptiv, implementează acțiuni de tip atomic.

Astfel, se elimină necesitatea implementării de tehnici pentru sincronizarea și controlul acceselor concurente la resursele partajate ale nucleului (inclusiv la variabilele globale). De asemenea, sunt evitate problemele de planificare de tip inversare a priorității.

Interfața cu utilizatorul

HARETICK pune la dispoziția utilizatorului/operatorului o interfață cu sistemul. Interfața este realizată printr-o serie de componente sistem care implementează un protocol simplu și flexibil de comunicații de date, un protocol de comandă și interogare a stării sistemului, și un set de proceduri de încărcare a unei aplicații noi în sistem.

3 Componentele HARETICK și relația dintre acestea

Nucleul de operare TR pentru sisteme încorporate HARETICK este conceput ca un set omogen de componente software de bază, care interacționează strâns unul cu celălalt, asigurând ca ansamblu, operarea platformei țintă (Motorola DSP56307 Evaluation Module) conform principiilor TR vizate.

Arhitectura generală a nucleului HARETICK (vezi Figura 3-1) este proiectată ținând cont de caracteristica esențială a operării sistemului – existența celor două contexte de operare: contextul lejer (preemptiv) și contextul strict (non-preemptiv). În consecință, toate elementele sistemului (task-urile sistem și ale aplicației) se vor clasifica după operarea într-unul din cele două contexte. Sistemul va cuprinde așadar, ModX-uri (task-uri TR stricte) și task-uri lejere.

3.1 Componentele HARETICK

Abordarea modulară a proiectării și implementării HARETICK se regăsește începând cu componentele de bază sistem – un set de task-uri speciale, sistem, cu rol de gestionare a resurselor hardware și software.

Fiecare componentă sistem efectuează o secvență bine definită de acțiuni esențiale în operarea corectă a HARETICK, incluzând:

- încărcarea și lansarea în execuție a sistemului;
- inițializarea sistemului;
- implementarea și gestionarea comunicației de comenzi și date cu un sistem de calcul gazdă, conectat la platforma țintă;
- interfașarea cu operatorul;
- încărcarea unei aplicații, de pe legătura de date cu sistemul gazdă, pentru execuție;
- planificarea task-urilor lejere;
- planificarea task-urilor stricte (ModX-urile);
- funcția de comutare a execuției task-urilor (executivul TR);
- generarea de diverse rapoarte de stare sistem și trimiterea lor către calculatorul gazdă; și așa mai departe.

În continuare vom prezenta pe scurt componentele principale ale HARETICK, specificând tipul lor (task-uri lejere sau ModX-uri) și funcțiile pe care le îndeplinesc. O reprezentare intuitivă a componentelor HARETICK, precum și a relațiilor stabilite între acestea, este redată în Figura 3-1.

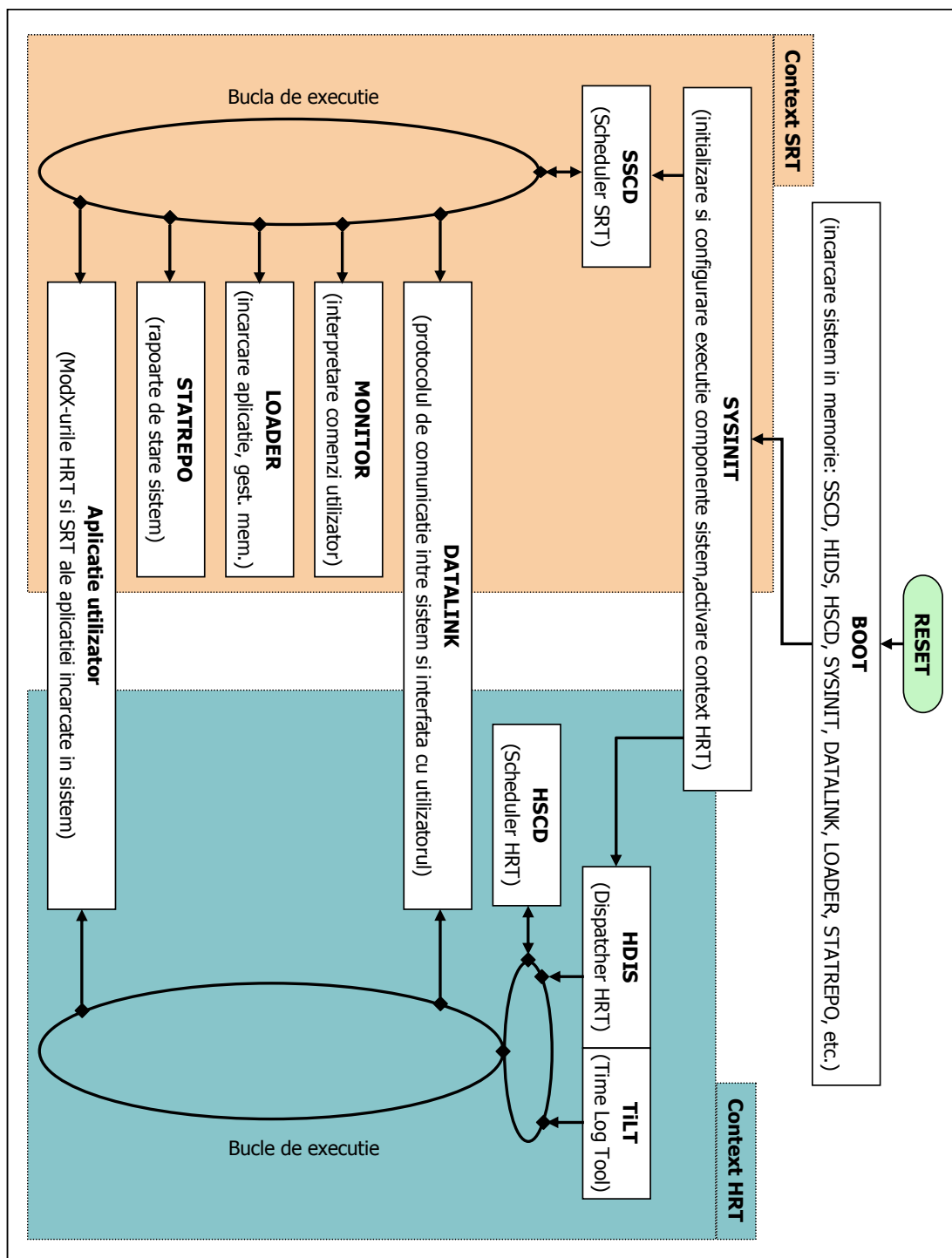


Figura 3-1. Componentele principale HARETICK și relația dintre ele

3.1.1 Încărcarea și lansarea sistemului: BOOT

Secvența (task-ul) de încărcare și startare a sistemului HARETICK este denumită "BOOT". Are rolul de a încărca în memorie structurile de date și codul tuturor componentelor de bază ale HARETICK: SYSINIT, SSCD, HDIS, TiLT, HSCD, DATALINK, MONITOR, LOADER, STATREPO, etc.

Task-ul BOOT este apelat și executat de către procesorul sistemului în momentul alimentării sau după o operație de RESET, și se găsește înscris în memoria EPROM internă sau externă a procesorului țintă.

Task-ul BOOT nu poate fi clasificat nici ca task lejer, nici ca ModX, deoarece în această etapă de operare a HARETICK, încă nu au fost inițiate cele două contexte de operare.

3.1.2 Inițializarea sistemului: SYSINIT

Task-ul de inițializare a sistemului este denumit "SYSINIT". Este task-ul lansat în execuție de către BOOT, după ce acesta a încărcat cu succes în memorie sistemul.

Rolul lui SYSINIT este de a inițializa parametrii de operare ai sistemului: variabilele globale sistem, variabilele și parametrii celorlate componente, cum ar fi pointerii către tabela cu descriptorii de procese și tabela de dispatch, etc. După efectuarea inițializărilor, SYSINIT activează contextul strict de execuție prin startarea ceasului TR (*RTC – Real-Time Clock*), care în mod automat, prin mecanismul întreruperilor de timer, apelează executivul sistemului, HDIS (Hard real-time Dispatcher).

SYSINIT este un task de tip lejer, deci practic, se poate considera că, odată cu lansarea în execuție a lui SYSINIT de către BOOT, se activează contextul lejer de execuție. La finalul secvenței lui SYSINIT este lansat planificatorul task-urilor lejere, SSCD, care gestionează în continuare operarea acestora în contextul lejer de execuție.

3.1.3 Comunicația de date: DATALINK

Comunicația de comenzi și date dintre nucleul HARETICK și mediul de dezvoltare INVERTA de pe un calculator gazdă este realizată cu ajutorul setului de task-uri DATALINK, care implementează o interfață simplă și flexibilă de comunicații bidirecționale cu următoarele caracteristici:

- (→) Încărcarea de comenzi utilizator, ce urmează apoi a fi interpretate și executate de către HARETICK;
- (→) Încărcarea de aplicații și date în sistem;
- (←) Descărcarea de rapoarte legate de starea curentă de execuție a sistemului;
- (←) Descărcarea periodică de rapoarte de timp.

În notațiile de mai sus a fost folosit simbolul (→) pentru a indica un sens al comunicației dinspre calculatorul gazdă către platforma țintă (cu HARETICK), iar (←) pentru sensul invers.

Grupul de task-uri DATALINK este compus atât din ModX-uri, cât și din task-uri lejere (vezi și Figura 3-1). ModX-urile DATALINK sunt responsabile pentru implementarea operațiilor de transmisie/recepție a biților de date pe interfața de comunicații a sistemului. Aceste operații sunt de natură periodică și strictă, trebuind să respecte o rată (tact de bit) bine stabilită. ModX-urile DATALINK se vor executa așadar în contextul strict, asigurând respectarea debitului stabilit pentru comunicație (respectarea ratei de comunicație). Ele vor fi planificate și executate în mod continuu, periodic, chiar dacă, la un moment dat, nu există informație ce trebuie transmisă/recepționată.

Task-urile lejere DATALINK au ca scop prelucrarea la un nivel mai superior a informațiilor vehiculate pe legătura de date a sistemului. Ele utilizează zone de memorie tampon (buffere) de transmisie/recepție pentru a dezasambla/asambla cuvintele de date vehiculate în timpul comunicațiilor. Task-urile lejere DATALINK se vor executa în contextul corespunzător, putând fi astfel întrerupte de către ModX-urile ce rulează în cadrul sistemului. Execuția task-urilor lejere DATALINK nu prezintă un tipar periodic și nici specificații stricte, ele fiind apelate de către alte task-uri sistem ce necesită la un moment dat utilizarea legăturii de date cu calculatorul gazdă.

3.1.4 Interfațarea cu operatorul: MONITOR

Task-ul MONITOR are rolul de interpretor de comenzi utilizator, transmise de către mediul de dezvoltare INVERTA de pe calculatorul gazdă, prin intermediul interfeței de comunicații.

De asemenea, MONITOR determină lansarea acțiunilor de încărcare a unei aplicații, descărcarea raportului de stare sistem, etc., prin invocarea task-urilor sistem corespunzătoare.

MONITOR este un task de tip lejer, implementând interacțiuni cu utilizatorul ce nu sunt critice pentru operarea sistemului țintă.

3.1.5 Încărcarea unei aplicații: LOADER

Încărcarea unei aplicații în sistemul țintă este realizată prin intermediul task-ului LOADER, care este invocat de către MONITOR ca urmare a interpretării comenzii aferente primite de la operator. Utilizând interfața de comunicații DATALINK, task-ul LOADER recepționează de la mediul de dezvoltare INVERTA de pe calculatorul gazdă toate datele și secvențele de cod ce definesc noua aplicație, printre care:

- tabela cu identificatorii de proces (PDT – Process Descriptor Table) corespunzătoare aplicației, în care se regăsesc principalii parametri ai task-urilor, incluzând pe cei ce descriu comportarea în timp (e.g. perioada, intervalul limită, durata maximă de execuție – WCET, contorul de execuții, și așa mai departe);
- graful dependențelor de program (PDG – Program Time Graph) ale task-urilor aplicației (descrie, cu ajutorul unei structuri unice de graf aciclic, direcționat, dependențele de control și de date);
- codul task-urilor aplicației;

- listele parametrilor de intrare, de ieșire și a celor globali pentru fiecare task;
- alte attribute ale task-urilor.

După recepționarea aplicației prin DATALINK, LOADER creează structurile de date și tabelele aferente, și configurează memoria sistemului prevăzută pentru încărcarea aplicației:

- înscrie codul fiecărui task în zona de memorie corespunzătoare codului aplicației (zona de text a aplicației, "CODE");
- creează zonele de memorie corespunzătoare parametrilor de ieșire ai task-urilor ("Data Out");
- generează tabela de simboluri a aplicației, în care se face corespondența dintre numele fiecărui parametru de intrare sau ieșire al task-urilor, cu adresa din memorie unde este localizat; etc.

LOADER are așadar și rolul de gestionare a memoriei, respectiv a structurilor din memoria sistemului ținută, rezervate aplicațiilor. Datorită faptului că HARETICK implementează un nucleu de operare pentru sisteme încorporate (embedded systems), pe de o parte zona structurilor de memorie proprii nucleului este gestionată practic de elementele acestuia și este configurată din etapa de proiectare și implementare, iar pe de altă parte nu există mecanisme care să controleze/restricționeze accesul task-urilor la diversele zone de memorie – acest lucru fiind responsabilitatea programatorului de sistem/aplicație.

La finalul secvenței LOADER, aplicația încărcată cu succes este lansată în execuție. Task-urile lejere ale aplicației vor fi planificate și activate de către planificatorul contextului lejer de execuție (SSCD), iar ModX-urile aplicației vor fi planificate de către planificatorul contextului strict de execuție (HSCD), fiind activate de către executivul sistemului (HDIS). Momentul lansării în execuție a aplicației este considerat ca fiind startul operării TR a aplicației în cadrul sistemului.

LOADER este un task de tip lejer, întregul set de operații, de la activarea și până la lansarea acestuia în execuție fiind considerat în afara contextului de operare timp-real a sistemului. Până la finalizarea încărcării și configurării noii aplicații în cadrul sistemului, HARETICK continuă să execute vechea aplicație, în regim timp-real ce respectă toți parametrii de comportare temporală specificați pentru aceasta.

3.1.6 Planificatorul task-urilor TR lejere: SSCD

Task-ul SSCD (Soft real-time Scheduler) are rolul de planificare a task-urilor lejere ale sistemului și aplicației, pentru execuție în context lejer pe platforma HARETICK.

SSCD este la rândul său un task lejer, execuția sa fiind întreruptă când este cazul de către ModX-urile aparținând contextului strict al sistemului.

3.1.7 Planificatorul ModX-urilor: HSCD

Task-ul HSCD (Hard real-time Scheduler) are rolul de planificare a task-urilor TR stricte (ModX-uri) ale sistemului și aplicației încărcate pe acesta, pentru execuție în contextul strict al HARETICK.

HSCD gestionează o tabelă de dimensiuni fixe (Tabela de Dispatch) în care înregistrează pe rând, ModX-urile ce urmează a fi executate în ciclul curent de planificare în contextul strict al HARETICK. O înregistrare din tabelă conține identificatorul ModX-ului corespunzător și instanța timpului sistem (valoarea RTC – ceasul timp-real) la care este planificată execuția acestuia.

Un ciclu de planificare este definit de terminarea execuției curente a HSCD și până la terminarea execuției ultimului ModX înscris în Tabela de Dispatch, atunci când, practic este apelat din nou HSCD.

Pe lângă înregistrările normale, pentru ModX-urile sistem, Tabela de Dispatch conține o înregistrare specială (prima înregistrare din tabelă), ce identifică pe HSCD.

HSCD este un ModX, fiind astfel executat de către HARETICK în contextul strict de operare.

3.1.8 Comutarea execuției ModX-urilor: HDIS

Comutarea execuției ModX-urilor în cadrul contextului strict al HARETICK este gestionată de către executivul TR al sistemului, HDIS (Hard real-time Dispatcher).

HDIS utilizează Tabela de Dispatch gestionată de HSCD, pe care o accesează în citire, înregistrare cu înregistrare. Accesarea a tabelii se face în mod circular, astfel încât după lansarea în execuție a ModX-ului referit de ultima înregistrare din tabelă, HDIS va reveni la baza tabelii, unde găsește înregistrarea specială corespunzătoare lui HSCD, declanșând astfel un nou ciclu de planificare în cadrul contextului strict.

Un alt rol important al HDIS este acela de salvare/restaurare a contextului task-urilor lejere a căror execuție este întreruptă de către contextul strict de operare a nucleului.

HDIS este apelat în mod automat în cadrul sistemului HARETICK, de către întreruperea de ceas timp-real (RTC), la momentul de timp ce specifică planificarea unui ModX. Odată activat, HDIS efectuează următoarele acțiuni:

- verifică dacă a fost întrerupt contextul lejer. În caz afirmativ, se efectuează salvarea contextului de execuție lejeră a task-ului întrerupt într-o zonă dedicată;
- identifică ModX-ul ce urmează a fi lansat în execuție la momentul curent, din Tabela de Dispatch;
- citește următoarea înregistrare din Tabela de Dispatch, corespunzând ModX-ului ce urmează ca planificare celui curent, și extrage momentul de start al execuției acestuia;
- programează timerele ceasului de timp-real (RTC) pentru a genera întrerupere la momentul de start al ModX-ului următor;
- decrementează contorul de execuții al ModX-ului curent (dacă acesta are valoare finită);
- trece ModX-ul curent din starea "planificat" ("SCD"), fie în starea "de execuție" ("RUN") – dacă valoarea contorului de execuție era nenulă înaintea decrementării, fie în starea "ModX fantomă" ("GST");
- dacă ModX-ul curent este în starea "RUN", îi cedează procesorul și îl apelează pentru execuție.

La terminarea execuției ModX-ului curent, acesta apelează din nou HDIS. În această situație HDIS efectuează următoarele acțiuni:

- trece ModX-ul curent din starea "de execuție" (RUN) sau "ModX fantomă" (GST), înapoi în "planificat" (SCD) sau în "gata de planificare" (RDY);
- avansează referința (pointerul) de citire al Tabelei de Dispatch către ModX-ul următor;
- verifică dacă există suficient timp pentru a comuta contextul strict cu cel lejer, permițând execuția task-urilor lejere ale sistemului și aplicației. Dacă nu este timp suficient, HDIS va iniția o buclă infinită în care va aștepta apariția întreruperii RTC corespunzătoare planificării ModX-ului următor. Dacă se decide comutarea controlului către contextul lejer, se va restaura contextul task-ului lejer a cărui execuție a fost întreruptă.

Din cele de mai sus rezultă că HDIS are două componente: HDIS Prefix (HDIS_PRE) care este apelat de întreruperea RTC și lansează în execuție ModX-uri, și HDIS Suffix (HDIS_SUF) care este apelat la terminarea fiecărui ModX.

HDIS este, la rândul său, un ModX, fiind astfel executat în context strict în cadrul HARETICK.

3.1.9 Raportarea stării curente a sistemului: STATREPO

Task-ul de raportare a stării curente a sistemului este denumit STATREPO. Rolul acestuia este de a trimite către interfața utilizator de pe calculatorul gazdă (INVERTA), rapoarte la cerere, utilizând interfața de comunicație DATALINK. Conținutul rapoartelor generate de STATREPO cuprinde:

- momentul curent al generării raportului;
- tabela cu identificatorii de proces (PDT – Process Descriptor Table) corespunzătoare aplicației executate curent, în care se regăsesc principalii parametri ai task-urilor, incluzând pe cei ce descriu comportarea în timp (e.g. perioada, intervalul limită, durata maximă de execuție – WCET, contorul de execuții, și așa mai departe);
- graful dependențelor de program (PDG – Program Time Graph) ale task-urilor aplicației (descrie, cu ajutorul unei structuri unice de graf aciclic, direcționat, dependențele de control și de date);
- codul task-urilor aplicației;
- listele parametrilor de intrare, de ieșire și a celor globali pentru fiecare task;
- alte attribute ale task-urilor;
- harta memoriei;
- conținutul curent al Tabelei de Dispatch;
- tabela de simboluri a aplicației; etc.

Așa cum a fost descris, în cadrul referatului precedent, sistemul OPEN-HARTS împreună cu componenta sa dedicată dezvoltării și analizei aplicațiilor TR stricte INVERTA, informațiile legate de starea curentă a sistemului HARETICK și a aplicației ce rulează pe acesta sunt necesare mediului INVERTA pentru analiza unei noi aplicații sau pentru modificarea celei curente.

STATREPO mai poate fi configurat pentru a genera și transmite rapoarte periodice privind timpii importanți ai execuției sistemului (istoria execuției).

STATREPO este un task de tip lejer, fiind activat doar la cererea operatorului și putând fi întrerupt de către contextul strict de execuție al HARETICK de câte ori este necesar.

3.1.10 Subrutina de raportare a evenimentelor sistem: TiLT

TiLT ("Time Log Tool") este o subrutină aparținând task-ului TR strict (ModX-ului) HDIS. TiLT este apelată atât de HDIS_PRE înainte de lansarea în execuție a unui ModX, cât și de HDIS_SUF la terminarea execuției ModX-ului curent.

Rolul TiLT este de a stoca într-o zonă dedicată de memorie tampon circulară (buffer circular), informații legate de evenimentele lansării sau terminării execuției ModX-urilor sistemului:

- valoarea ceasului timp-real (RTC) în momentul apariției evenimentului;
- starea HDIS din acel moment;
- valoarea contorului de execuție.

Fiind o subrutină a ModX-ului HDIS, TiLT este executat, la rândul său, în context strict, non-preemptiv.

3.2 Relația dintre componentele nucleului HARETICK

Figura 3-1 reprezintă intuitiv atât componentele de bază ale nucleului TR HARETICK, cât și relația dintre acestea. În figură sunt evidențiate cele două contexte de execuție ce constituie caracteristica distinctivă a HARETICK: contextul strict (non-preemptiv) în cadrul căruia operează ModX-urile sistemului și ale aplicației (task-urile TR stricte), și contextul lejer (preemptiv) în care operează task-urile lejere.

Relația dintre componentele HARETICK poate fi prezentată și prin scenariul ce presupune la un moment dat pornirea platformei țintă (sistemul de control încorporat) ce operează sub nucleul HARETICK, după care se încarcă în sistem o aplicație oarecare pentru a fi executată. Scenariul evidențiază următoarele evenimente:

1. În urma startării platformei țintă (sau după o operație de RESET), execuția procesorului este redirectată către zona de start a codului secvenței BOOT din memoria EPROM a sistemului.
Odată activat, task-ul BOOT încarcă în memoria sistemului (RAM) structurile de date și codul celorlalte componente ale nucleului, după care lansează inițializarea sistemului (apelează task-ul SYSINIT).
2. În faza de inițializare a sistemului, task-ul lejer SYSINIT setează valorile parametrilor de operare ai nucleului, variabilele globale, parametrii și variabilele celorlalte componente ale nucleului.
Lansarea task-ului SYSINIT corespunde practic cu activarea contextului lejer de operare al HARETICK. După inițializarea parametrilor sistemului,

SYSINIT configurează și pornește ceasul de timp-real (RTC) care, prin intermediul întreruperii de timer apelează executivul TR al HARETICK, HDIS. Prin aceasta, contextul strict de operare al sistemului este, la rândul lui, activat.

3. Odată activat contextul strict, execuția SYSINIT este întreruptă de către HDIS, care identifică din Tabela de Dispatch, ModX-ul care urmează a fi lansat în execuție. În această fază de operare a sistemului, singurul ModX prezent în tabelă este planificatorul pentru contextul strict, HSCD. Prin urmare, HSCD este apelat de către HDIS.

Înainte de a ceda procesorul către HSCD, executivul HDIS apelează subrutina de raportare a evenimentelor sistem, TiLT, pentru a marca lansarea în execuție a lui HSCD.

4. Planificatorul de ModX-uri, HSCD, va aplica algoritmi de planificare non-preemptivă special concepuți, asupra setului de ModX-uri existente în momentul curent în cadrul sistemului (referiți în PDT – tabela cu descriptorii de proces). Planificarea ModX-urilor durează până în momentul în care HSCD a completat toate înregistrările din Tabela de Dispatch cu indentificatorii și momentele de start corespunzătoare ciclului curent de planificare.

Ca observație, menționăm ca HSCD va modifica și înregistrarea proprie din Tabela de Dispatch, setând pentru momentul de start, noua valoare la care este planificat HSCD după execuția ultimului ModX în cadrul ciclului de planificare. Cu alte cuvinte, după execuția curentă a HSCD, prima înregistrare din Tabela de Dispatch va avea setată valoarea corespunzătoare momentului de start al ciclului de planificare următor.

5. Contextul de execuție strictă se va continua după terminarea execuției planificatorului HSCD, prin lansarea celorlalte ModX-uri din Tabela de Dispatch la momentele de timp corespunzătoare.

Printre aceste ModX-uri, aparținând configurației inițiale de operare a nucleului HARETICK, se regăsesc ModX-urile componentei DATALINK, ce asigură comunicația de comenzi și date cu un calculator gazdă.

În Figura 3-1 sunt reprezentate două bucle de execuție pentru contextul strict, subliniind faptul că fiecare ModX este lansat în execuție de către HDIS și că planificarea acestora se face ciclic, pe baza Tabelei de Dispatch, de către HSCD care se execută odată la fiecare ciclu.

6. În intervalele de timp neocupate de execuția contextului strict, se vor executa task-urile contextului lejer. Prima dată când controlul execuției revine contextului lejer, se continuă execuția task-ului SYSINIT care termină inițializările celorlalte task-uri (aparținând contextului lejer) și lansează planificatorul task-urilor lejere, SSCD.
7. SSCD are rolul de a planifica, în intervalele de timp neocupate de execuția contextului strict al HARETICK, task-urile lejere ale sistemului și ale aplicației (după ce se încarcă o anumită aplicație în sistem). În această fază a scenariului, task-urile lejere considerate de algoritmul de planificare preemptivă al SSCD

sunt: task-urile lejere ale DATALINK, task-ul MONITOR, LOADER și STATREPO. Acestea sunt planificate și executate după un algoritm de planificare de tip "time-sharing", preemptiv, pe bază de priorități.

În Figura 3-1 este reprezentată bucla de execuție corespunzătoare contextului lejer al HARETICK, definită de task-ul de planificare SSCD.

4 Reprezentarea și gestionarea timpului

4.1 Timpul ca parametru esențial de operare al sistemului

Timpul, ca și coordonată esențială a sistemelor TR stricte, trebuie să fie implicat în toate fazele de dezvoltare a acestora: specificare și verificare formală, programare, analiză, planificare și execuție, și implementare (așa cum subliniază referatul de doctorat anterior).

În consecință, întreaga concepție și implementare a nucleului HARETICK ține cont de acest obiectiv principal.

Funcționarea sistemelor de calcul în general, și a sistemelor TR în particular, se bazează pe o frecvență de tact generată de un oscilator intern sau extern. Frecvența de tact definește așa-numitul *ciclu procesor* sau *ciclu instrucțiune* al unității de procesare a sistemului, reprezentând durata execuției unei instrucțiuni procesor (exprimată în secunde, Hz sau perioade de tact). Totodată, frecvența de tact alimentează baza de timp a sistemului, anume ceasul de timp-real (*RTC – Real-Time Clock*). Prin urmare, granularitatea de timp minimă cu care sistemul poate lucra este egală cu durata unui ciclu procesor.

Utilizarea RTC, a cărei arhitectură include un contor de timp, definește domeniul temporal de operare al sistemului. Astfel,

$$t_0 = 0$$

poate fi identificat cu momentul startării sistemului (cuplarea la tensiune sau după o operație de RESET, urmate de inițializarea contorului RTC), fiind denumit *momentul inițial*.

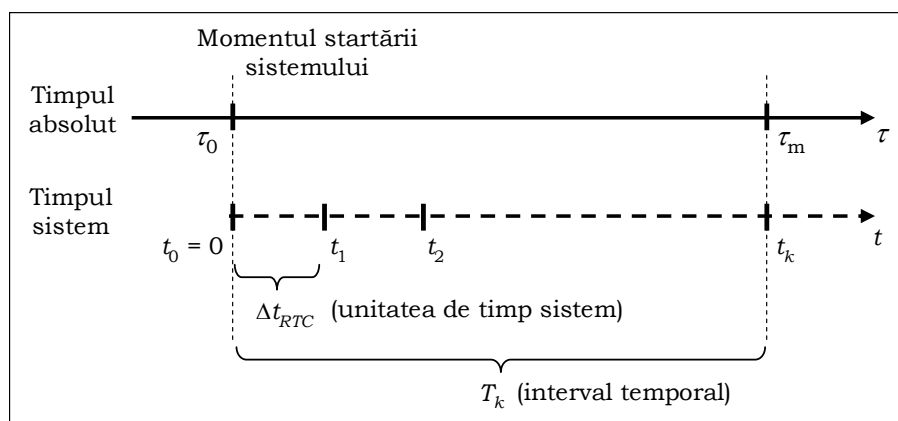


Figura 4-1. Timpul sistem și timpul absolut

În Figura 4-1 sunt reprezentate cele două domenii temporale: timpul sistem și cel absolut, la care se raportează primul. Variabilele de timp au fost notate distinct, τ pentru timpul absolut și t pentru timpul sistem. La momentul τ_0 are loc startarea sistemului (deci, $t_0 = 0$).

Oricare două instanțe succesive ale timpului sistem, t_i și t_{i+1} sunt distanțate cu Δt_{RTC} în domeniul temporal absolut, și cu 1 în domeniul temporal al sistemului. Două evenimente care apar în intervalul Δt_{RTC} sunt văzute ca fiind simultane de către sistem. De aici rezultă valoarea unității temporale sistem.

Lungimea unui interval temporal oarecare T_k poate fi determinată astfel:

$$\begin{cases} T_k = \tau_m - \tau_0 = k \cdot \Delta t_{RTC} & \text{pentru timpul absolut} \\ T_k = t_k - t_0 = k & \text{pentru timpul sistem} \end{cases} \quad (4-1)$$

Legătura dintre cele două domenii temporale este sintetizată de relația de corespondență a unei instanțe de timp (un punct din cele două domenii):

$$\tau_i = \tau_0 + (t_i - t_0) \cdot \Delta t_{RTC} \quad (4-2)$$

În cele ce urmează vom discuta aspectele legate de reprezentarea și gestionarea timpului în cadrul nucleului TR HARETICK pentru sisteme de control încorporat și DSP. Acolo unde se impun particularizări, se vor exemplifica abordările din cadrul implementării HARETICK pe platforma Motorola DSP56307EVM.

Frecvența tactului de core a procesorului DSP56307 a fost setată la valoarea:

$$CoreCLK = 32 \text{ MHz.}$$

În cazul acestei arhitecturi, ciclul procesor ce definește execuția instrucțiunilor este egal cu o perioadă de tact, rezultând astfel granularitatea de timp minimă ca fiind egală cu durata ciclului procesor:

$$\Delta t_{CLK} = 31,25 \text{ ns.}$$

Pe de altă parte, se va utiliza un ceas de timp-real (RTC) ce aplică tactului de core al DSP un factor de divizare (prescalare) egal cu 4. Rezultă pentru unitatea de timp sistem (unitatea de timp măsurată de RTC), valoarea:

$$\Delta t_{RTC} = 4 \cdot \Delta t_{CLK} = 125 \text{ ns.}$$

Astfel, HARETICK poate determina evenimentele din cadrul sistemului cu o frecvență maximă de $8 \text{ MHz} = CoreCLK / 4$, echivalentă cu 4 cicli procesor.

Observație

Frecvența de lucru a procesorului și unitatea de timp sistem au fost stabilite la valorile de mai sus în implementarea HARETICK din considerente practice.

Astfel, pentru frecvența de lucru a DSP s-a căutat o valoare comodă din punctul de vedere al calculelor legate de timp. Inițial a fost încercată programarea blocului PLL (care generează practic frecvența de lucru internă a DSP pe baza unui oscilator extern, de frecvență $F_{ext} = 12.288 \text{ MHz}$), pentru valoarea $CoreCLK = 80 \text{ MHz}$. Arhitectura internă și parametrii de operare ai PLL nu au permis însă o astfel de setare.

Unitatea de timp sistem Δt_{RTC} a fost stabilită la 4 cicli instrucțiune din considerente legate de relaxarea preciziei cu care se calculează timpii de

execuție ai ModX-urilor, oferindu-se astfel un interval de toleranță destinat acoperii eventualelor erori și imprecizii ce derivă din calculul acestor timpi.

4.2 Structurile destinate reprezentării și gestionării timpului în HARETICK

Timpul sistem este reprezentat și gestionat în cadrul HARETICK utilizând un set de structuri și mecanisme hardware și software, ce vor fi descrise pe scurt în continuare.

4.2.1 Timpul absolut sistem

Nucleul HARETICK implementează o structură pentru gestionarea timpului absolut, utilizând timpul sistem intern și setări ale operatorului de sistem, provenite din exterior (de la interfața INVERTA), prin intermediul unor comenzi specifice.

Timpul absolut sistem este stocat într-o variabilă sistem, `Sys_RealTime`, de dimensiune egală cu 3 cuvinte procesor:

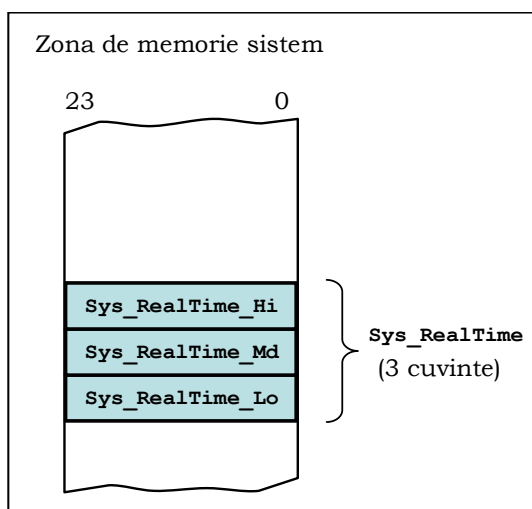


Figura 4-2. Reprezentarea timpului absolut sistem

Ținând cont că Motorola DSP56307 este o arhitectură pe 24 biți, timpul absolut sistem este exprimat pe 72 biți. În aceste condiții, și considerând setările pentru unitatea de timp sistem ($\Delta t_{RTC} = 125$ ns), timpul absolut pe care îl poate reprezenta și gestiona sistemul HARETICK se întinde pe o durată maximă:

$$\max \{ \text{Sys_RealTime} \} = 18.718.157,355 \text{ ani.}$$

Parametrul sistem `Sys_RealTime` poate fi setat (actualizat) de către operator prin intermediul unor comenzi specifice, lansate din mediul INVERTA, la valoarea timpului curent sau după necesități. În timpul operării sistemului HARETICK, `Sys_RealTime` este actualizată în mod constant, ciclic, de către ModX-ul HSCD

(planificatorul contextului strict de execuție). Actualizarea timpului absolut sistem se face în cadrul fiecărei execuții a HSCD, cu valoarea acumulată a intervalului de timp corespunzător ultimului ciclu de planificare.

4.2.2 Ceasul timp-real

Ceasul timp-real (RTC) este practic elementul activ de gestionare a timpului în sistemul HARETICK. RTC are două roluri esențiale:

- (a) Contor de timp, utilizat pentru referiri ale componentelor sistemului la instanța curentă de timp sistem. Orice task, fie că aparține nucleului HARETICK, fie aplicației ce rulează la un moment dat în sistem, poate citi valoarea curentă a contorului RTC, având astfel la dispoziție o bază de timp real. Fiind vorba de un sistem TR (strict) destinat aplicațiilor critice, accesul oricărui task la baza de timp a sistemului reprezintă o facilitate extrem de utilă și necesară.

- (b) Singura sursă (predictibilă) de întreruperi.

Așa cum a fost subliniat în cadrul referatului anterior, abordarea pe care o propunem și o adoptăm în ceea ce privește proiectarea și dezvoltarea sistemelor TR stricte presupune restricționarea pe cât posibil a mecanismului de întreruperi.

În implementarea curentă a nucleului HARETICK, ceasul timp-real reprezintă singura sursă de întreruperi din cadrul sistemului. Timerele RTC sunt programate de către executivul TR al nucleului, HDIS, pentru a genera câte o întrerupere în momentul corespunzător lansării în execuție a fiecărui ModX, conform planificării calculate de HSCD și înscrise în Tabela de Dispatch.

Mai mult, RTC reprezintă o sursă predictibilă de întreruperi, în sensul că se poate determina oricând, și de către orice task, momentul apariției următoarei întreruperi. Acest lucru se poate realiza în două moduri: (i) prin citirea valorii înscrise la un moment dat în registrii de comparare ai RTC, sau (ii) prin citirea înregistrării din Tabela de Dispatch, corespunzătoare ModX-ului ce urmează a fi executat după cel curent. Înregistrarea respectivă conține informația de timp referitoare la momentul începerii execuției ModX-ului, corespunzând următoarei întreruperi RTC.

Pentru a ușura operarea și utilizarea ceasului timp-real, este de dorit o variantă bazată pe un contor de timp cu capacitate cât mai mare. Implementarea curentă a sistemului HARETICK se bazează pe arhitectura procesorului Motorola DSP56307, care prevede existența a 3 timere interne, pe 24 biți fiecare. Astfel, RTC este implementat cu ajutorul a două timere cascade, capacitatea totală de contorizare a acestuia fiind prin urmare de 48 biți.

Figura 4-3 prezintă arhitectura RTC din HARETICK, bazată pe cascada a două timere pe 24 biți fiecare, *Timer0* și *Timer1*, cu numărare crescătoare. Ca observație, menționăm că DSP56307 pune la dispoziția programatorului de sistem/aplicație și un al treilea timer, *Timer2*, care poate fi utilizat pentru diverse alte contorizări în cadrul aplicațiilor.

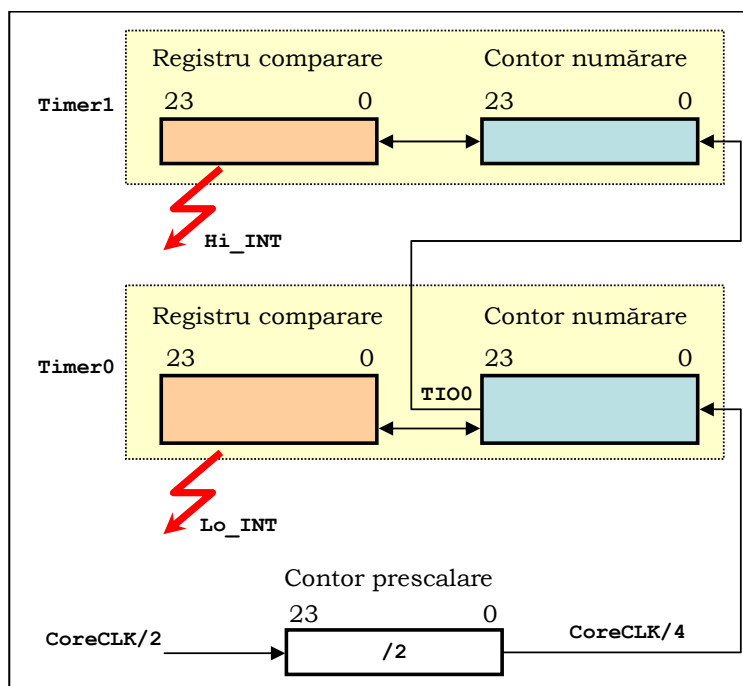


Figura 4-3. Ceasul timp-real (RTC) în sistemul HARETICK

În aceste condiții, și considerând setările pentru unitatea de timp sistem ($\Delta t_{RTC} = 125 \text{ ns}$), timpul sistem pe care îl poate reprezenta și gestiona RTC se întinde pe o durată maximă:

$$\max \{RTC\} = 407,226 \text{ zile},$$

iar capacitatea maximă a contorului Timer0 al RTC este de:

$$\max \{Timer0\} \cong 2 \text{ sec.}$$

Cascadarea celor două timere ale DSP56307 pentru obținerea unei capacități suficiente de contorizare a timpului real complică mecanismul de întreruperi al HARETICK, în sensul că introduce două întreruperi RTC în loc de una: Lo_INT, și Hi_INT. Întreruperile se activează atunci când valoarea contorului de numărare ajunge la valoarea din registrul de comparare corespunzător, pentru fiecare timer în parte. Mecanismul de întreruperi RTC va fi detaliat în cadrul prezentării executivului TR al HARETICK, HDIS, acesta fiind ModX-ul care lucrează în mod direct cu ceasul timp-real (Capitolul 7).

Gestionarea timpului în cadrul RTC începe în momentul activării contextului strict de execuție al HARETICK, operație inițiată de către task-ul SYSINIT. În acel moment valoarea timpului real sistem este:

$$t_{RTC} = t_0 = 0$$

Pe lângă executivul TR al nucleului (HDIS), ceasul timp-real mai este utilizat de către planificatorul contextului strict, HSCD, pentru calcularea și exprimarea timpilor de start ai ModX-urilor planificate. HSCD nu folosește însă în mod direct RTC și

nu-i modifică valoarea, ci utilizează structuri proprii pentru exprimarea timpului de planificare. Acestea vor fi descrise în paragraful următor.

4.2.3 Timpul de planificare

Planificatorul contextului strict de execuție al HARETICK, anume ModX-ul HSCD, gestionează, la rândul său, timpul, dispunând de structuri și mecanisme proprii pentru aceasta.

Așa cum a fost detaliat în referatul anterior, algoritmi de planificare propuși calculează planificarea ModX-urilor pe un interval de timp de forma:

$$T_{SCHED} = [0, T_{LCM}),$$

unde T_{LCM} reprezintă cel mai mic multiplu comun al perioadelor ModX-urilor din setul supus planificării.

În consecință, ModX-ul de planificare HSCD, utilizează așa-numitul *timp absolut de planificare*, t_{SCHED} , care este definit pe intervalul $[0, T_{LCM})$. Dat fiind faptul că lungimea intervalului de planificare (valoarea lui T_{LCM}) variază puternic cu perioadele și numărul ModX-urilor, putând ajunge la valori foarte mari, în implementarea HARETICK se utilizează o variabilă de capacitate suficientă (vezi Figura 4-4).

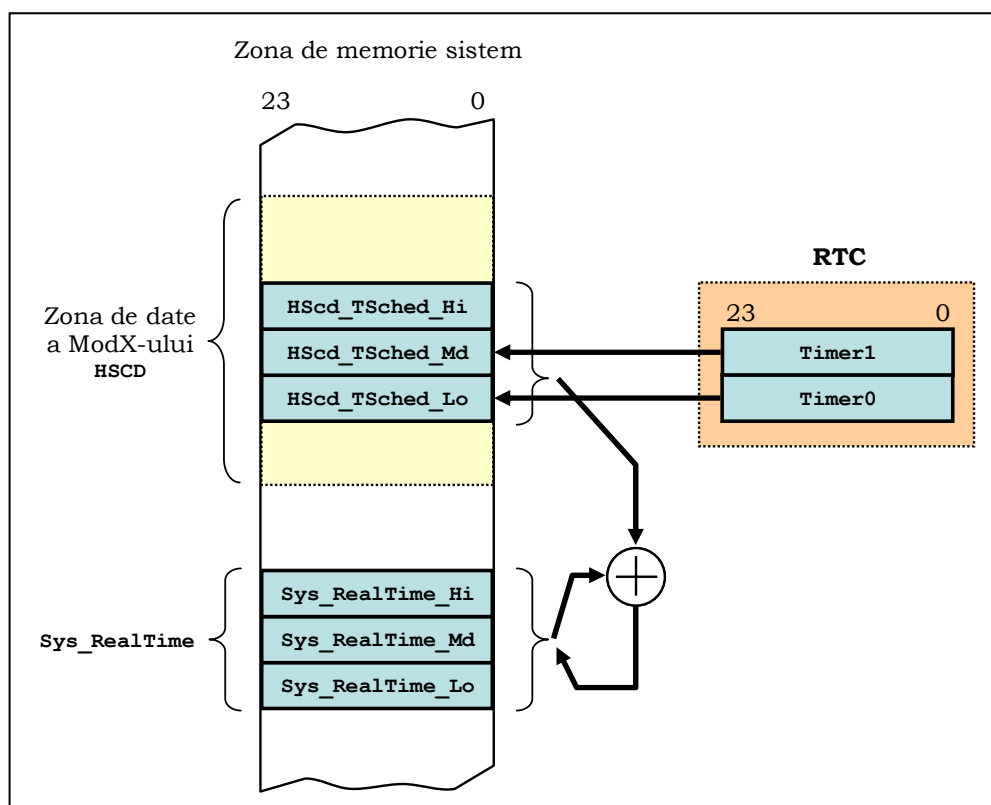


Figura 4-4. Structurile și mecanismele de gestionare a timpului absolut de planificare

Cum rezultă și din figură, timpul absolut de planificare (sau, simplu – *timpul de planificare*) este exprimat pe 3 cuvinte a câte 24 biți, rezultând o capacitate totală de 72 biți (identică cu cea corespunzătoare timpului absolut sistem). Figura 4-4 indică și faptul că timpul de planificare este sincronizat cu cele două timere ale RTC.

Timpul de planificare calculat de către algoritmul de planificare implementat de HSCD este apoi utilizat pentru completarea înregistrării corespunzătoare din Tabela de Dispatch. În tabelă, HDIS va înscrie, pe lângă identificatorul ModX-ului planificat, și valoarea timpului real sistem la care va fi lansat în execuție ModX-ul respectiv de către HDIS. Astfel, componenta temporală a unei înregistrări din Tabela de Dispatch va avea valoarea `HScd_TSched_Md:HScd_TSched_Lo`.

4.3 Mecanismele de sincronizare a structurilor de gestionare a timpului în HARETICK

Relațiile dintre cele trei tipuri de timpi utilizați în cadrul sistemului HARETICK, adică: timpul absolut sistem, timpul real sistem (RTC) și timpul de planificare, sunt bine definite:

- Există o singură structură activă de generare și măsurare a timpului în cadrul sistemului: RTC, care gestionează timpul real sistem – baza de timp a sistemului.
- Timpul real sistem (t_{RTC}) este generat (modificat) doar de timerele procesorului și este referit de componentele de bază ale nucleului HARETICK: HDIS, HSCD, etc. El poate fi de asemenea citit de către alte task-uri (stricte sau lejere) aparținând nucleului sau aplicației curente.
- Inițializarea RTC este efectuată de către task-ul SYSINIT, o singură dată, la pornirea sistemului
- Timpul absolut sistem poate fi setat și referit la nivelul operatorului, prin intermediul unor comenzi specifice, și nu este utilizat de nucleu la operații de sincronizare a execuțiilor sau pentru planificarea ModX-urilor.
- ModX-ul de planificare a contextului strict de execuție, HSCD, utilizează timpul de planificare (t_{SCHED}), sincronizându-l cu timpul real sistem (RTC) la fiecare execuție. La începutul fiecărui ciclu de planificare, cu ocazia lansării sale în execuție, HSCD va efectua următoarele acțiuni:
 - Verifică relația de corespondență dintre timpul de planificare și RTC, și în caz că e necesar, efectuează sincronizarea:

$$\begin{cases} \text{HScd_TSched_Md} \leftarrow \text{Timer1} \\ \text{HScd_TSched_Lo} \leftarrow \text{Timer0} \end{cases}$$

- Sincronizează timpul absolut sistem cu timpul de planificare:

$$\begin{cases} \text{Sys_RealTime_Hi} \leftarrow \text{Sys_RealTime_Hi} + \text{HScd_TSched_Hi} \\ \text{Sys_RealTime_Md} \leftarrow \text{Sys_RealTime_Md} + \text{HScd_TSched_Md} \\ \text{Sys_RealTime_Lo} \leftarrow \text{Sys_RealTime_Lo} + \text{HScd_TSched_Lo} \end{cases}$$

- Aplică algoritmul de planificare non-preemptivă a setului de ModX-uri existent în sistem, pentru completarea tuturor înregistrărilor din Tabela de Dispatch. Astfel, dacă la momentul curent al planificării, $t_{SCHED,i}$, a fost selectat pentru execuție ModX-ul M_j , înregistrarea curentă din Tabela de Dispatch va conține următoarele elemente (MSW seminifică cel mai semnificativ cuvânt, iar LSW – cel mai puțin semnificativ):

$$\left\{ \begin{array}{l} 1. \text{ Identificatorul ModX - ului : } M_j \\ 2. \text{ MSW al momentului planificării } \left(t_{st}^{M_j} \right) : (t_{SCHED,i}).HScd_TSched_Hi \\ 3. \text{ LSW al momentului planificării } \left(t_{st}^{M_j} \right) : (t_{SCHED,i}).HScd_TSched_Lo \end{array} \right.$$

- În momentul umplerii Tabelei de Dispatch, HSCD completează înregistrarea specială din tabelă ce face referire el (prima înregistrare), cu timpul de start al următoarei execuții ale sale – specificând începutul unui nou ciclu de planificare.
- Momentele de start ale execuției ModX-urilor planificate în cadrul unui ciclu de planificare vor fi referite, pe rând, din Tabela de Dispatch de către executivul nucleului, HDIS. Înaintea lansării ModX-ului curent, HDIS programează registrele de comparare ale RTC cu cele două cuvinte (MSW, respectiv LSW) ale timpului de start corespunzător ModX-ului ce urmează celui curent.

Operațiile referitoare la timp, desfășurate de componentele HARETICK, vor fi detaliate în cadrul secțiunilor ce descriu componentele respective în cadrul lucrării.

5 Reprezentarea ModX-urilor în cadrul sistemului

5.1 Modelul task-ului TR strict: ModX-ul

Unul din rezultatele preocupărilor noastre în ceea ce privește studiul proiectării și implementării sistemelor TR stricte pentru aplicații critice, îl reprezintă modelul task-urilor TR stricte, ModX-ul.

Un *ModX* (*Modul executabil*) reprezintă un task TR periodic, modular, cu specificații complete și stricte de comportare în timp și cu planificare și execuție în context non-preemptiv, destinat a servi ca element de bază în specificarea, proiectarea și dezvoltarea sistemelor și aplicațiilor TR stricte, pentru care determinismul și predictibilitatea operării sunt cerințe esențiale.

Din perspectiva arhitecturii aplicațiilor TR stricte, cât și din punct de vedere funcțional, ModX-ul este un modul software, adică echivalentul unui "bloc de bază" (*basic block*) utilizat în teoria analizei programelor și a compilării. ModX-ul prezintă un set de intrări (parametri de intrare), un corp (bloc) de instrucțiuni care prelucrează intrările, variabilele definite local și variabilele globale, și un set de ieșiri (parametri de ieșire). Din punct de vedere al specificațiilor și al comportamentului în domeniul timp, parametrii de intrare/ieșire ai ModX-ului sunt semnale interne ale sistemului. În plus, un ModX poate interacționa cu semnale de intrare (ModX-ul *tratează* semnale de intrare) și de ieșire (ModX-ul *generează* semnale de ieșire).

Formal, ModX-ul este o componentă a unei aplicații TR, reprezentate sub forma unui graf de program, direcționat, aciclic:

$$\mathbf{G} \equiv \langle \mathbf{M}, \mathbf{V} \rangle \quad (5-1)$$

unde $\mathbf{M} = \{M_i \mid 1 \leq i \leq |\mathbf{M}|\}$ este mulțimea nodurilor grafului $\mathbf{G}$ (ModX-urile sistemului) și $\mathbf{V} = \{(M_i, M_j) \mid M_i, M_j \in \mathbf{M}; 1 \leq i, j \leq |\mathbf{M}|\}$ este mulțimea arcelor lui $\mathbf{G}$, constând din perechi ordonate de noduri.

Din punct de vedere formal, un ModX M_i este reprezentat prin următorul model:

$$M_i \equiv \langle \mathbf{T}, \mathbf{P}, \mathbf{S}, \mathbf{F} \rangle \quad (5-2)$$

unde:

$\mathbf{T} = \{T_{pr}^{M_i}, T_{ex}^{M_i}, T_{dl}^{M_i}, T_{dy}^{M_i}, N^{M_i}\}$ este setul specificațiilor de comportare temporală a ModX-ului M_i ,

$\mathbf{P} = \{\mathbf{P}_{IN}, \mathbf{P}_{OUT}, \mathbf{P}_{GLB}\}$ este setul parametrilor de intrare, de ieșire, respectiv, al variabilelor globale cu care operează M_i ,

$\mathbf{S} = \{\mathbf{S}_{IN}, \mathbf{S}_{OUT}\}$ este setul semnalelor de intrare tratate de M_i și al semnalelor de ieșire generate de M_i , iar

F este setul de instrucțiuni care procesează informația în cadrul lui M_i , reprezentând specificarea funcțională a ModX-ului.

Setul T al specificațiilor temporale ale ModX-ului M_i se compune din următoarele elemente:

- $T_{pr}^{M_i}$ (*period*) este perioada lui M_i – caracteristică esențială și obligatorie pentru toate ModX-urile, fiind task-uri periodice. Perioada este constantă de-a lungul operării lui M_i ;
- $T_{ex}^{M_i}$ (*execution time*) este durata de execuție a lui M_i . Este de asemenea constantă pe durata operării sistemului TR, fiind egală cu *WCET* al codului ModX-ului (*Worst Case Execution Time*);
- $T_{dl}^{M_i}$ (*deadline*) este intervalul limită pentru planificarea și execuția lui M_i în cadrul perioadei acestuia. $T_{dl}^{M_i}$ este un parametru opțional, introdus pentru cazurile în care este necesar ca execuția lui M_i să se finalizeze cu un interval de timp oarecare înainte de terminarea perioadei, pentru toate ciclurile de execuție;
- $T_{dy}^{M_i}$ (*delay*) este intervalul de întârziere a planificării și execuției lui M_i în cadrul perioadei. $T_{dy}^{M_i}$ este, la rândul lui un parametru opțional, ce poate fi utilizat în cazurile în care se dorește ca planificarea și execuția lui M_i să nu poată fi efectuate într-un anumit interval de la începutul fiecărei perioade a acestuia;
- N^{M_i} (*numărul de execuții, execution count*) este numărul total de execuții ale lui M_i . Este un parametru opțional, care tratează trei cazuri:
 - M_i are un număr nedefinit de execuții (execuție continuă): $N^{M_i} = \infty$,
 - M_i are un număr finit, bine precizat de execuții: $N^{M_i} < \infty$. În acest caz, M_i interacționează cu semnale (generează sau tratează semnale) cu apariție temporară,
 - M_i nu mai este executat de către sistem: $N^{M_i} = 0$. M_i este planificat în continuare, dar nu va mai fi lansat în execuție de către executivul sistemului.

Observație

În cazul în care N^{M_i} este finit, valoarea momentană a acestuia poate fi controlată din două perspective:

- i.) În timpul execuției lui M_i , executivul sistemului de operare decrementează de fiecare dată pe N^{M_i} cu o unitate, până acesta ajunge la 0. În

acest mod, sistemul contorizează execuțiile lui M_i pe parcursul operării. Când N^{M_i} ajunge la 0, executivul nu-l va mai lansa în execuție pe M_i .

ii.) În timpul execuției unui alt ModX, M_j , acesta poate accesa contorul de execuții al lui M_i , reactualizându-l la orice valoare posibilă, conform necesităților aplicației. Prin acest mecanism, se poate relansa în execuție un ModX, chiar dacă acesta are contorul setat în mod curent pe 0.

Figura 5-1 exemplifică operarea pentru două perioade consecutive a unui ModX oarecare M_i , parametrii săi temporali fiind reprezentați corespunzător.

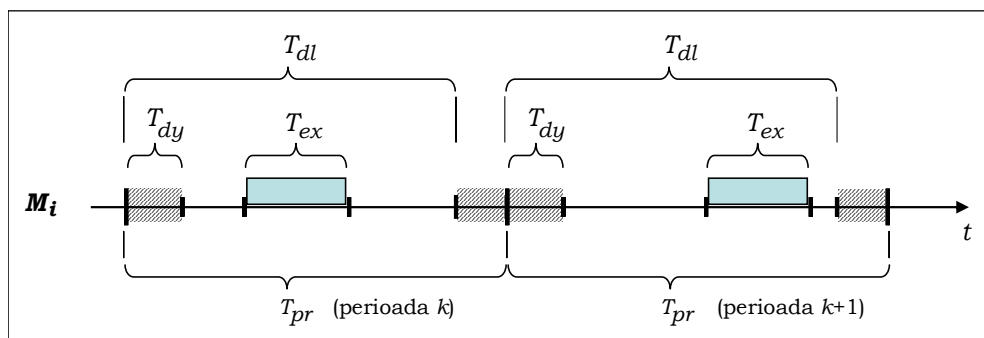


Figura 5-1. ModX-ul M_i și parametrii săi temporali

Modelul de ModX propus în cadrul abordării de față prezintă o serie de proprietăți importante, tratate în referatul anterior. Din perspectiva implementării ModX-urilor în cadrul nucleului HARETICK, subliniem următoarele aspecte:

1) *Abordarea modulară*

ModX-urile reprezintă module de sine stătătoare, ce încorporează secvențe proprii de cod, seturi de parametri de intrare și de ieșire și accesează variabile globale sistem sau ale aplicației.

Pentru fiecare din aceste componente ale ModX-urilor este necesară implementarea unor mecanisme specifice în cadrul nucleului HARETICK.

2) *Interacțiunea cu semnalele sistemului*

ModX-urile pot genera semnale de ieșire sau pot trata semnale de intrare. Setul semnalelor cu care interacționează un ModX este stabilit în faza de specificare a ModX-ului (setul S din (5-2)).

Parametrii temporali ai semnalelor de intrare vor influența comportarea în timp a ModX-ului (parametrii setului T din (5-2)), iar parametrii semnalelor de ieșire sunt determinați de comportarea temporală a ModX-ului.

3) *Planificarea și execuția ModX-urilor*

ModX-urile sunt task-uri TR stricte, periodice, planificate și executate în regim non-preemptiv.

Contextul execuției ModX-urilor în cadrul sistemului este denumit *contextul strict de execuție* al HARETICK, fiind prevăzute mecanisme speciale pentru operarea acestui context.

Pe de altă parte, task-urile lejere ale sistemului/aplicației (soft real-time tasks) sunt planificate și executate în regim preemptiv, în cadrul unui context separat – *contextul lejer de execuție* al HARETICK. Contextul lejer de execuție a fost introdus în modelul sistemului TR strict pe care îl avem în vedere, din două considerente principale:

- Aplicațiile ce rulează pe sisteme TR, chiar cele critice din punct de vedere al operării, includ și task-uri ce nu necesită asigurarea și garantarea unei comportări exacte în timp. Aceste task-uri reprezintă modelul task-urilor lejere considerate în abordarea de față.
- Planificarea non-preemptivă a ModX-urilor în cadrul contextului strict de execuție implică o reducere semnificativă a eficienței și performanțelor sistemului. Prin introducerea task-urilor lejere, planificate și executate într-un context complementar – contextul lejer, a fost vizată creșterea eficienței sistemului.

Contextul strict de execuție are precedență asupra celui lejer în cadrul nucleului HARETICK.

4) *Comunicația inter-procese și sincronizarea acceselor la resurse partajate*

Fiind module de sine stătătoare, executate în context non-preemptiv, ModX-urile implementează practic operații atomice. De aici rezultă eliminarea necesității unor mecanisme de sincronizare a acceselor concurente la resursele partajate ale sistemului sau ale aplicației.

De asemenea, se elimină problemele de tip "inversare a priorității" (*priority inversion*), tipice sistemelor TR care implementează tehnici de planificare și execuție preemptive, bazate pe priorități. În HARETICK, toate ModX-urile, ce se execută în contextul strict, non-preemptiv, au aceeași prioritate – maximă; planificarea ordinii de execuție a acestora fiind rezolvată cu algoritmi special concepuți.

Prin concepția modulară, a fost asigurat și un mecanism simplu de comunicare între ModX-uri: transmiterea de parametri de intrare/ieșire și, în plus, există posibilitatea utilizării (limitate) a variabilelor globale.

5.2 Reprezentarea ModX-urilor în cadrul HARETICK

Nucleul de operare HARETICK implementează un set de structuri și mecanisme destinate reprezentării, planificării și execuției ModX-urilor. Dat fiind tipul sistemului considerat – nucleu de operare TR strict pentru platforme bazate pe DSP sau platforme de control digital încorporat (embedded systems), reprezentarea ModX-urilor se face pe baza unei arhitecturi pe cât posibil statice.

ModX-urile gestionate la un moment dat de nucleul HARETICK se împart în două categorii: ModX-uri sistem și ModX-uri aplicație. HARETICK utilizează

structuri distincte pentru cele două categorii de ModX-uri, deși din punctul de vedere al tipului de sistem implementat, această diferențiere nu este obligatorie (din acest motiv nu a fost prevăzut nici un mecanism de control al accesului task-urilor la memoria sistemului, de exemplu; rezolvarea acestei chestiuni fiind responsabilitatea programatorului de sistem/aplicație).

ModX-urile sistem sunt încărcate de către task-ul BOOT la pornirea platformei, având alocate un set de resurse distincte (de exemplu, în cazul memoriei, există o zonă dedicată ModX-urilor sistem – zona sistem). ModX-urile sistem vor fi discutate separat în cadrul lucrării de față, făcând parte integrantă din nucleul HARETICK.

În continuare vom discuta cazul general al unui ModX ce aparține unei aplicații oarecare, încărcate pe platforma țintă ce operează sub nucleul HARETICK.

O primă structură implicată în gestionarea ModX-urilor aplicației este tabela ce conține graful direcționat aciclic de program al aplicației (*PDAGT – Program Directed Acyclic Graph Table*). Fiecare înregistrare a tabeli constituie o reprezentare a unui nod al grafului (adică a unui ModX al aplicației):

- identificatorul ModX-ului: *PID*;
- numărul de nivel în cadrul grafului;
- numărul de părinți ai nodului;
- referință către părintele (părinții) nodului;
- numărul de fii ai nodului;
- referință către fiul (fiii) nodului.

Până în momentul de față a fost studiată reprezentarea aplicațiilor ce rezultă în grafuri de program de tip arbore (fiecare nod are un singur părinte și unul sau mai mulți fii). Ca preocupare de viitor în cadrul programului de doctorat, avem în vedere conceperea de structuri (statice) care să poată reprezenta grafuri de program complexe (un nod poate avea mai mulți părinți și mai mulți fii).

Tabela PDAGT este creată de către task-ul LOADER al nucleului HARETICK, în faza de încărcare a aplicației în cadrul sistemului țintă, prin intermediul interfeței de comunicație cu mediul INVERTA, implementată de DATALINK. Translatarea structurii aplicației, dintr-o reprezentare dinamică de tip graf, într-una statică de tip tabelă este posibilă datorită operațiilor de analiză a aplicației dezvoltate în mediul INVERTA, pe calculatorul gazdă. Astfel, printre informațiile suplimentare furnizate de INVERTA la încărcarea aplicației în sistemul HARETICK, se găsește și un set de parametri ce definesc cantitativ graful aplicației (numărul total de noduri – ModX-uri, numărul de nivele ale grafului, etc.).

5.2.1 Reprezentarea ModX-urilor în memorie

Tot în cadrul operațiilor de încărcare a aplicației în sistem, task-ul LOADER alocă și completează, pentru fiecare ModX, un set de structuri de memorie, într-o zonă dedicată: zona aplicație.

Pentru fiecare ModX, sunt prevăzute următoarele structuri de memorie:

- Zona de cod (text), în care se încarcă secvența de cod compilat a ModX-ului;
- Zona parametrilor de ieșire, ce conține valorile fiecărui parametru de ieșire al ModX-ului.

De asemenea, pentru întreaga aplicație, este alocată o zonă de memorie pentru variabilele globale, și o zonă de memorie comună pentru parametri de intrare și variabilele locale ale ModX-urilor, denumită *Heap*.

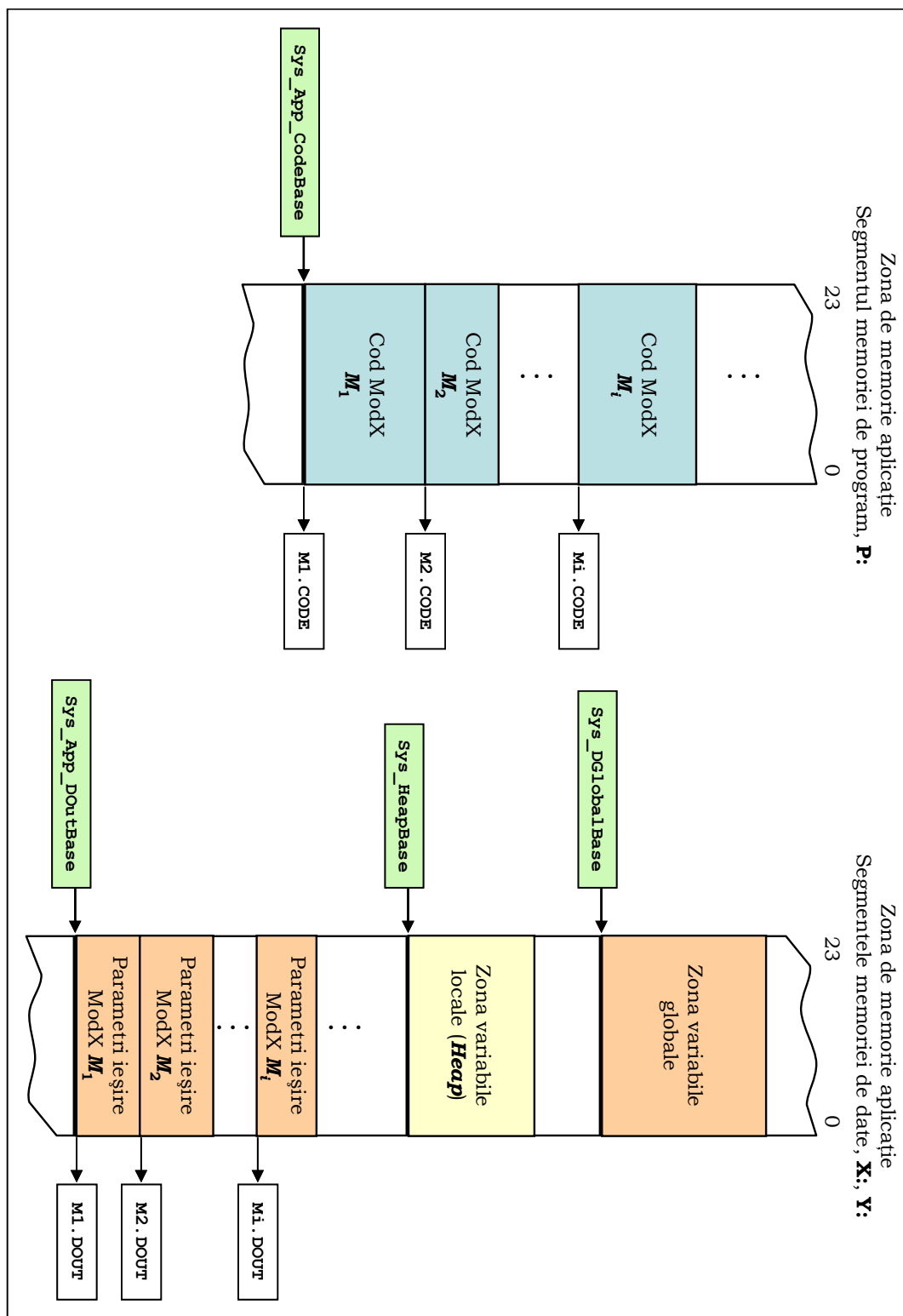


Figura 5-2. Reprezentarea ModX-urilor în memorie

La fel ca în cazul tabelii PDAGT, toate structurile utilizate de sistem pentru reprezentarea în memorie a ModX-urilor sunt definite și alocate static, pe baza informațiilor suplimentare legate de dimensiunile maxime ale fiecărei structuri, primite de HARETICK de la mediul de dezvoltare INVERTA, la încărcarea aplicației în sistem. De exemplu, pentru zona *Heap*, LOADER-ul va alocă o zonă de memorie de dimensiune cel puțin egală cu cea maximă necesară oricărui ModX din cele ale aplicației pentru lucrul cu variabilele sale locale (și parametrii de intrare).

Figura 5-2 prezintă structurile necesare nucleului HARETICK pentru reprezentarea în memorie a ModX-urilor aplicației. Codul fiecărui ModX este înscris în segmentul de memorie program al DSP56307 (memoria **P:**), începând de la o adresă de bază, *Sys_App_CodeBase*, care constituie un parametru sistem. În același timp, LOADER-ul reține adresele de start ale zonei de cod pentru fiecare ModX (*Mi.CODE*), pentru a le înscrie în celelalte tabele care servesc nucleului la gestionarea ModX-urilor, și care vor fi discutate în paragrafele următoare.

Zona parametrilor de ieșire ai fiecărui ModX este alocată în segmentul de memorie de date al DSP56307 (memoria **X:** sau **Y:**), începând de la o adresă de bază, *Sys_App_DOutBase*, care este, de asemenea, un parametru sistem.

Zona *Heap* este alocată de către LOADER, pe considerentele de dimensiune prezentate mai sus, începând de la adresa *Sys_HeapBase*, și, în manieră similară se procedează cu zona variabilelor globale ale aplicației, folosind ca adresă de bază, *Sys_DGlobalBase*.

5.2.2 Tabela de simboluri

În aceeași fază de încărcare a aplicației în cadrul sistemului HARETICK, task-ul LOADER creează tabela de simboluri, în care este înscrisă corespondența dintre fiecare parametru sau variabilă a ModX-urilor și adresa de memorie alocată pentru aceasta.

Astfel, de exemplu, pentru primul parametru de ieșire al unui ModX oarecare M_i (adică pentru parametrul alocat la startul zonei *Mi.DOUT*), LOADER-ul va trece în tabela de simboluri numele parametrului și adresa corespunzătoare valorii *Mi.DOUT*.

Pentru parametrii de intrare sau variabilele interne ale unui ModX, tabela de simboluri va conține numele variabilei și deplasamentul stabilit de către LOADER în cadrul zonei *Heap* pentru variabila respectivă.

5.2.3 Tabela descriptorilor de procese

Tabela descriptorilor de procese (*PDT – Process Descriptor Table*) este creată și completată tot de către LOADER, în faza de încărcare a aplicației în sistem. PDT conține identificarea fiecărui ModX, împreună cu caracteristicile sale (atribute, parametri temporali, etc.).

Figura 5-3 prezintă structura PDT și a unei înregistrări ale acesteia, care are o dimensiune de 13 cuvinte (de 24 biți în cazul arhitecturii DSP56307) și descrie atributele și parametrii unui ModX oarecare. Semnificația elementelor înregistrării PDT este prezentată în Tabelul 5-1.

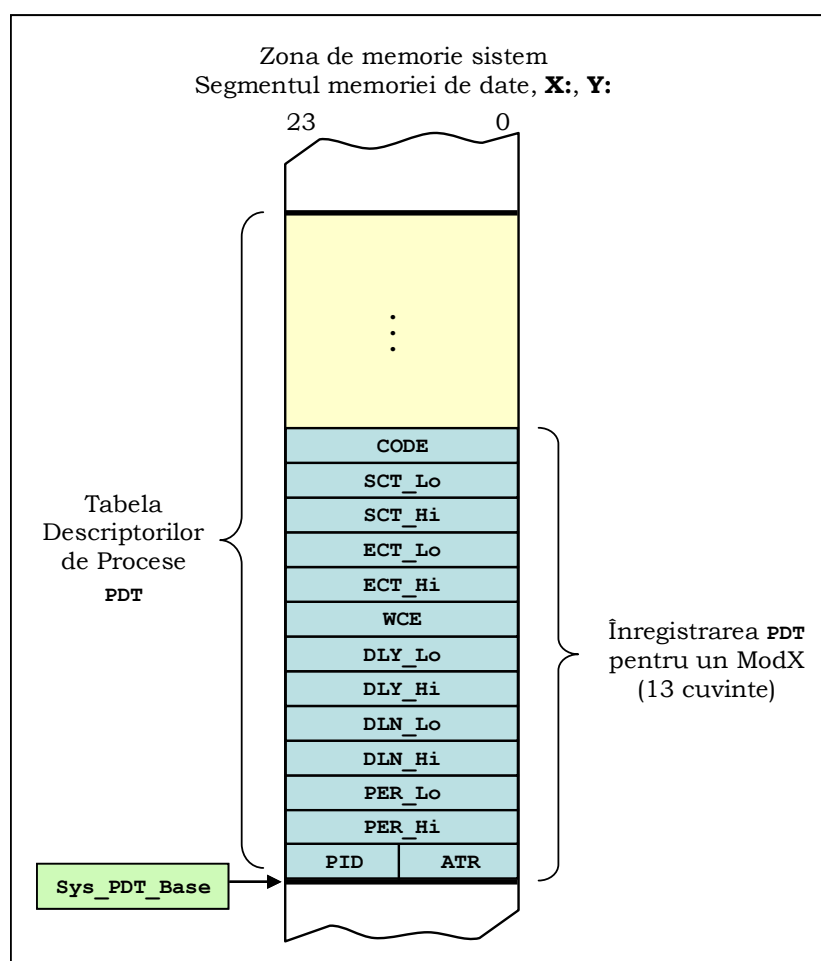


Figura 5-3. Tabela Descriptorilor de Procese (PDT)

Tabelul 5-1. Semnificația elementelor unei înregistrări din PDT

Offset	Denumire	Semnificație
0	PID (12 biți)	Identificatorul de proces, pentru ModX-ul M_i .
	ATR (12 biți)	Atribute suplimentare ModX (e.g. "FIXED", etc.).
1:2	PER_Hi:PER_Lo	Perioada ModX-ului (period), $T_{pr}^{M_i}$.
3:4	DLN_Hi:DLN_Lo	Intervalul limită (deadline), $T_{dl}^{M_i}$.
5:6	DLY_Hi:DLY_Lo	Intervalul de întârziere (delay), $T_{dy}^{M_i}$.
7	WCE	Durata maximă de execuție (execution time), $T_{ex}^{M_i}$.
8:9	ECT_Hi:ECT_Lo	Contorul de execuții (execution count), N^{M_i} .
10:11	SCT_Hi:SCT_Lo	Contorul de planificări (scheduling count) – utilizat de algoritmul de planificare al HSCD.
12	CODE	Adresa de start a zonei de cod a ModX-ului M_i .

Majoritatea parametrilor din PDT corespunzători unui ModX sunt utilizați de către HSCD, în cadrul algoritmului de planificare non-preemptivă a contextului strict de execuție. PDT este referită de asemenea de către executivul TR al HARETICK, HDIS, pentru identificarea ModX-ului ce urmează a fi lansat în execuție (`PID`), pentru citirea adresei de start a secvenței de cod a ModX-ului (`CODE`) și pentru accesarea (în citire-scriere) a contorului de execuții (`ECT`).

Din punct de vedere particular, al implementării HARETICK pe platforma Motorola DSP56307, structura PDT permite următoarele valori maxime pentru parametrii sistem și ai ModX-urilor:

- Dimensiune `PID`: 12 biți, rezultând un total de 4 K (4096) ModX-uri ce pot fi gestionate de către HARETICK;
- Dimensiune totală PDT: 100 înregistrări, rezultând un total de 100 ModX-uri ce pot fi gestionate de către sistem;
- Perioada unui ModX, `PER`, e exprimată pe 2 cuvinte (48 biți), rezultând un interval maxim de timp de 407,226 zile;
- Același interval maxim de timp poate fi exprimat pentru intervalul limită al planificării (`DLN`) și pentru intervalul de întârziere (`DLY`) ale ModX-ului: 407,226 zile;
- Durata maximă de execuție a ModX-ului (`WCE`) poate fi de maxim 2 sec (aproximativ), parametrul fiind exprimat pe un cuvânt procesor (24 biți);
- Contorul de execuții al ModX-ului (`ECT`) are o dimensiune de 2 cuvinte (48 biți), rezultând un număr maxim de peste $281 \cdot 10^{12}$ execuții ale ModX-ului ce pot fi gestionate de sistem;
- Contorul de planificări (`SCT`) poate exprima, la rândul lui, peste $281 \cdot 10^{12}$ planificări ale ModX-ului.

Atributul "FIXED" specificat în Tabelul 5-1 ca exemplificare pentru parametrul `ATR` al ModX-urilor se referă la cerința de execuție fixă a unui ModX în cadrul perioadelor acestuia (cu alte cuvinte, față de momentul de început al fiecărei perioade, ModX-ul respectiv se va executa cu o aceeași întârziere). Această problemă a fost tratată din punct de vedere teoretic și exemplificată, în referatul anterior.

5.3 Operarea de principiu a ModX-urilor în cadrul HARETICK

Modelul task-urilor TR stricte (ModX-urile) și întreaga concepție de reprezentare și operare a acestora în cadrul nucleului HARETICK au la bază cerința esențială de maximizare a predictibilității sistemului, cu garantarea respectării specificațiilor de comportare temporală a task-urilor stricte, oferindu-se astfel, o platformă sistem ce facilitează dezvoltarea, implementarea și operarea aplicațiilor critice de prelucrare de semnale și control digital încorporat.

Ca linii generale, pentru satisfacerea acestor cerințe, HARETICK utilizează structuri statice, limitate în timp și spațiu. Pentru fiecare aplicație încărcată în sistem se vor cunoaște parametrii maximi de operare ai acesteia: timpi, resurse sistem, memorie ocupată, etc.

Amintim, de asemenea, faptul că toate ModX-urile existente la un moment dat în sistem, operează în contextul strict, non-preemptiv de execuție, oferit de nucleul HARETICK.

În cele ce urmează, vom exemplifica operarea sistemului pentru cazul a două ModX-uri, M_i și M_j , care aparțin unei aplicații oarecare, încărcată pe platforma sistemului. Presupunem că M_i și M_j se află într-o relație de precedență de forma: $M_j \rightarrow M_i$, precedență rezultată ca urmare a dependenței de control și de date a aplicației. Figura 5-4 ilustrează acest scenariu, reprezentând cele două noduri (ModX-uri) ale grafului aplicației, împreună cu dependențele de program corespunzătoare. Fiecare nod al grafului va avea specificații complete: numele (M_j , M_i), parametrii de intrare (paranteza superioară din figură), parametrii de intrare (paranteza inferioară), parametrii de comportare temporală, etc. De exemplu, ModX-ul M_i are 3 parametri de intrare (param1, param2 și param3) și un parametru de ieșire (param 3). Primii doi parametri de intrare ai lui M_i provin de la parametrii de ieșire ai lui M_j (de unde rezultă și dependența de date dintre cele două), iar param 3 reprezintă în același timp, parametru de intrare și de ieșire, pentru M_i .

Posibilitatea ca un ModX oarecare să aibă parametri care constituie în același timp intrare și ieșire, derivă din modelul conceput pentru task-urile TR stricte. Acestea sunt module periodice, executate în context non-preemptiv, rezultând că prin acest mecanism ele își pot transmite informații de stare, de la o execuție la alta (adică de la o perioadă la alta de execuție).

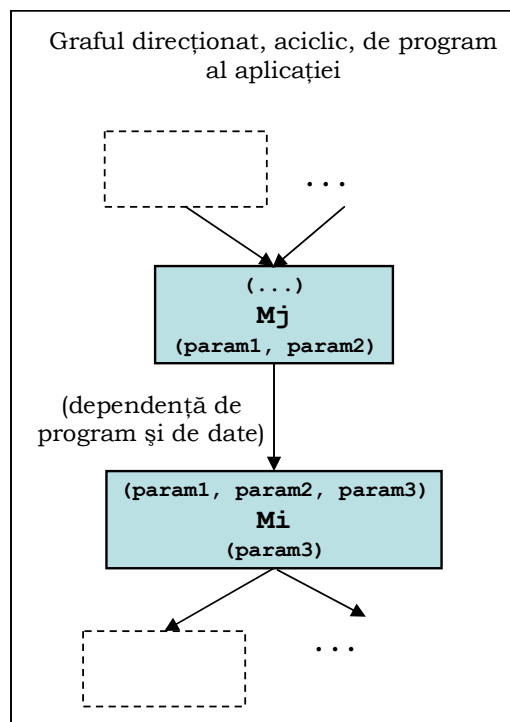


Figura 5-4. Exemplificare cu porțiunea de graf al aplicației ce conține cele două ModX-uri, M_i și M_j

Presupunem de asemenea, în cadrul acestui scenariu, că secvența de program a lui M_i lucrează cu un număr oarecare de variabile locale (de exemplu, 10), M_i ocupând, din acest punct de vedere, un spațiu maxim de memorie dintre toate ModX-urile aplicației. Următorii pași generali vor fi întâlniți în operarea nucleului HARETICK, cu privire la execuția celor două ModX-uri din Figura 5-4:

- 0) Aplicația a fost încărcată în sistem, structurile de date corespunzătoare au fost alocate și inițializate (PDAGT, PDT, zonele de cod și "Data Out" pentru M_j și M_i , *Heap*-ul pentru variabilele locale și parametrii de intrare, zona de variabile globale, tabela de simboluri, etc.).

Pentru zona de *Heap*, task-ul LOADER al nucleului va alocă, pentru toate ModX-urile aplicației, o dimensiune de memorie de 13 cuvinte: 10 cuvinte pentru variabilele locale ale lui M_i și 3 cuvinte pentru parametrii săi de intrare. Această decizie a LOADER-ului rezultă din ipoteza că M_i ocupă spațiul maxim de memorie pentru variabilele locale și parametrii de intrare, dintre toate ModX-urile aplicației.

În cadrul contextului strict de execuție, HSCD a planificat cele două ModX-uri, M_j și M_i , pentru execuție la momentele $t_{st,k}^{M_j}$, respectiv, $t_{st,m}^{M_i}$, unde k reprezintă perioada execuției lui M_j și m perioada execuției lui M_i . Ca rezultat al planificării, Tabela de Dispatch va avea o configurație de tipul celei din Figura 5-5. Ca observație, momentul de start al ModX-ului M_i nu poate apărea mai devreme de terminarea execuției lui M_j , adică:

$$t_{st,m}^{M_i} \geq t_{st,k}^{M_j} + T_{ex}^{M_j} = t_{st,k}^{M_j} + M_j \cdot WCE$$

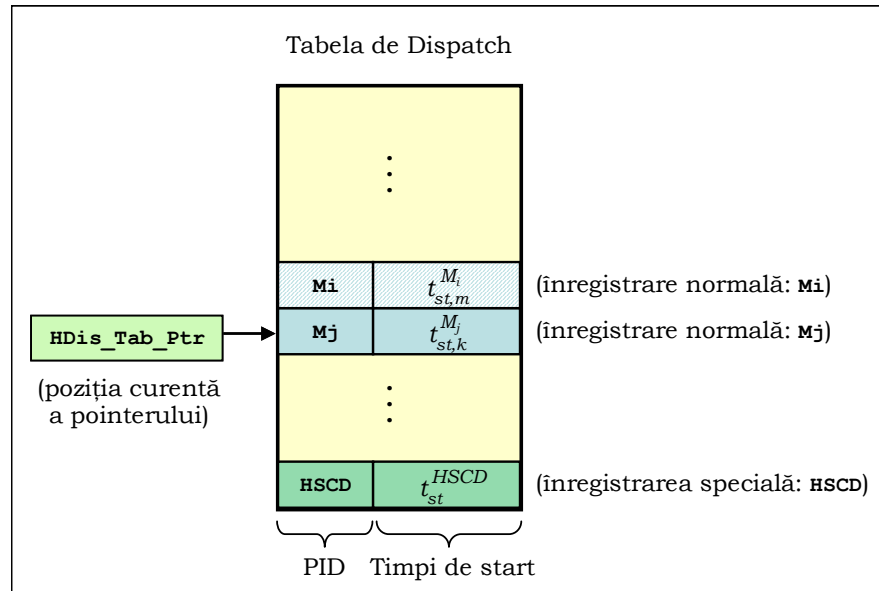


Figura 5-5. Configurația Tabelei de Dispatch pentru scenariul considerat

Presupunem că sistemul tocmai termină execuția ModX-ului M_j . Prin urmare, pointerul Tabelii de Dispatch ($HDis_Tab_Ptr$) indică spre M_j (ModX-ul curent în execuție).

- 1) Secvența de instrucțiuni a lui M_j se termină în mod obligatoriu cu apelarea componentei sufix a executivului TR al HARETICK, $HDis_SUF$ (vezi secțiunea 3.1.8 din Capitolul 3, și Capitolul 7). Instrucțiunea de apel la $HDis_SUF$ înlocuiește astfel, instrucțiunile de tip `return` sau `RTS` (Return from Subroutine), care încheie în mod uzual subrutinele de program (procedurile, funcțiile sau programele principale).

Ca urmare, se activează $HDis$, care actualizează pointerul Tabelii de Dispatch, $HDis_Tab_Ptr$, astfel încât acesta va indica spre M_i . De asemenea, $HDis_SUF$ verifică dacă mai este timp suficient, din momentul curent (al terminării execuției efective a lui M_j) până la momentul $t_{st,m}^{M_i}$, pentru a reda controlul contextului lejer de execuție. În caz contrar, $HDis_SUF$ inițiază o buclă de așteptare a întreruperii RTC pentru $t_{st,m}^{M_i}$.

- 2) La momentul $t_{st,m}^{M_i}$, RTC activează întreruperea de timer, având ca rezultat lansarea lui $HDis_PRE$ în execuție. Acesta programează regiștrii de comparare ai timerelor RTC pentru momentul startării execuției ModX-ului ce urmează lui M_i ca planificare, în Tabela de Dispatch. În continuare $HDis_PRE$ verifică valoarea contorului de execuții ($M_i.ECT$) al lui M_i , existând trei posibilități:

- (2.1) $M_i.ECT = ECT_CONT$: ModX-ul M_i are specificat (momentan) un număr continuu de execuții. Ca rezultat, $HDis$ nu va modifica acest parametru și va lansa în execuție pe M_i .
- (2.2) $M_i.ECT$ are o valoare finită. În acest caz, $HDis$ va decrementa cu o unitate contorul de execuții și apoi va lansa în execuție pe M_i .
- (2.3) $M_i.ECT = 0$: ModX-ul M_i este un *ModX fantomă* (vezi și discuția din referatul anterior). Ca rezultat, $HDis$ lasă nemodificat contorul de execuții și nu va lansa în execuție pe M_i , ci va apela pe $HDis_SUF$ (ca și când s-ar fi terminat execuția lui M_i).

- 3) Odată activat ModX-ul M_i , înainte de a începe execuția instrucțiunilor din zona de cod proprie, acesta va copia în locațiile din zona Heap corespunzătoare parametrilor săi de intrare, valorile parametrilor de ieșire ai ModX-urilor față de care M_i prezintă dependențe de date. Referirea parametrilor de ieșire se face pe baza tabelii de simboluri a HARETICK, care indică pentru fiecare parametru, adresa din zona "Data Out" corespunzătoare ModX-ului de care aparține.

Astfel, în scenariul de față, M_i va copia în cele trei locații de memorie asignate de LOADER în Heap pentru parametrii săi de intrare ($Heap_param1$, $Heap_param2$ și $Heap_param3$), valorile $M_j.param1$, $M_j.param2$, respectiv, $M_i.param3$. Cele trei valori sunt regăsite de M_i , prin referirea la zona $M_j.DOut$

pentru primii doi parametri și $M_i.DOut$ pentru cel de-al treilea, utilizând adresele corespondente din tabela de simboluri a HARETICK.

Observație

Motivația acestui mecanism de copiere a parametrilor de intrare ai unui ModX înaintea execuției secvenței sale propriu-zise de cod, derivă din necesitatea obținerii unei predictibilități maxime de operare a sistemului HARETICK.

Cum numărul parametrilor de intrare/ieșire diferă de la un ModX la altul într-un mod transparent executivului TR al sistemului, a-l implementa pe acesta să efectueze operațiile de copiere a valorilor momentane, sau utilizarea unei stive, reprezintă soluții extrem de ineficiente și surse de impredictibilitate majoră în cadrul sistemului.

Pe de altă parte, includerea acestor operații în codul ModX-urilor, cu calcularea corespunzătoare a duratei maxime de execuție ($M_i.WCE$) astfel încât să cuprindă și timpii necesari copierii parametrilor, conferă sistemului o mai mare flexibilitate, și, mai ales, *nu îi afectează predictibilitatea*.

Operațiile de copiere a valorilor parametrilor de intrare pentru un ModX se pot specifica, fie explicit – prin scrierea secvenței respective de instrucțiuni chiar la începutul codului program al ModX-ului, fie implicit, de către compiler (de exemplu) – prin inserarea instrucțiunilor la începutul secvenței de cod compilat al ModX-ului.

Din cele de mai sus se desprinde și motivația structurilor de memorie care reprezintă zonele cu parametrii de ieșire ai fiecărui ModX, "Data Out", propuse în proiectul HARETICK. Mai mult, cu ajutorul acestor structuri se garantează păstrarea valorilor pe care un ModX dorește să le comunice (altor ModX-uri, sau lui însuși), de la execuția curentă, până la următoarea execuție (perioadă) a sa.

- 4) Execuția lui M_i se continuă după copierea parametrilor săi de intrare în Heap, cu secvența de cod corespunzătoare, stocată în zona de cod ($M_i.CODE$) de către LOADER, și referită în cadrul înregistrării corespunzătoare din tabela descriptorilor de procese (PDT).

La sfârșitul execuției, parametrul `param3`, alocat în zona "Data Out" a lui M_i ($M_i.param3$) va conține noua valoare, rezultată în urma execuției lui M_i . Execuția lui M_i se finalizează prin apelarea directă a lui `HDIS_SUF`.

5.4 Concluzii

Task-urile TR stricte sunt modelate, în concepția lucrării de față, prin ModX-uri – task-uri periodice, modulare, cu specificații complete și stricte de comportare în timp și cu planificare și execuție în context non-preemptiv. Un ModX conține seturi de parametri de intrare și de ieșire, variabile globale, un set de parametri ce îi specifică comportarea în timp, seturile semnalelor de intrare și de ieșire cu care interacționează, și secvența de cod program (specificarea funcțională).

În cadrul nucleului HARETICK, ModX-urile sunt reprezentate cu ajutorul unor structuri și mecanisme specifice:

- "PDAGT" (Program Directed Acyclic Graph Table), tabela ce conține graful direcționat, aciclic, de program al aplicației;
- Zona de cod ("CODE") a ModX-ului;
- Zona parametrilor de ieșire ("Data Out");
- Zona variabilelor globale ale aplicației;
- Zona variabilelor locale și ai parametrilor de intrare ai ModX-ului în execuție ("Heap");
- Tabela de simboluri;
- Tabela descriptorilor de procese ("PDT").

ModX-urile sunt executate în cadrul contextului strict al HARETICK, fiind planificate de către ModX-ul sistem HSCD. Activarea și terminarea ModX-urilor este gestionată de către executivul TR al nucleului, HDIS.

La începutul execuției fiecărui ModX, are loc în mod obligatoriu operația de copiere a valorilor parametrilor de ieșire, din zonele "Data Out" ale ModX-urilor de care depinde cel curent, în locațiile parametrilor de intrare, prevăzute în "Heap". Timpul de execuție a instrucțiunilor implicate de această operație se vor adăuga la WCET-ul ModX-ului.

Comunicația inter-task-uri este realizată prin transmiterea parametrilor de intrare/ieșire, cu mecanismul descris anterior. Datorită structurii și reprezentării ModX-urilor (care, practic, implementează operații atomice), și a mecanismului de comunicație dintre ele, controlul și sincronizarea acceselor la resurse concurente nu mai este o problemă. Un alt aspect interesant legat de acest subiect îl constituie facilitatea de transmitere a stării unui ModX, de la o execuție a sa la alta, prin intermediul unui același parametru care este și de intrare, și de ieșire.

La sfârșitul execuției fiecărui ModX, se găsește instrucțiunea de apel la HDIS, care, practic înlocuiește în mod echivalent instrucțiunile utilizate uzual în acest caz (`return`, `exit`, `end`, `RTS`, etc.).

Din punctul de vedere al cerințelor impuse de arhitectura nucleului HARETICK, calcularea timpului maxim de execuție a ModX-urilor (WCET) implică următoarele elemente:

- durata de execuție a celei mai lungi căi de program din codul propriu-zis al ModX-ului;
- durata secvenței de instrucțiuni, de la începutul execuției ModX-ului, care efectuează operațiile de copiere a parametrilor de ieșire;
- durata instrucțiunii de apel la `HDIS_SUF`, de la sfârșitul secvenței de cod a ModX-ului.

6 Funcționarea de principiu a sistemului

6.1 Contextele de execuție sistem și comutarea acestora

Conceptul abordat în cadrul activității noastre de doctorat, referitor la sistemele TR stricte destinate aplicațiilor critice de achiziție și prelucrare numerică a semnalelor și de control încorporat (embedded systems), se bazează pe următoarea idee de bază:

Utilizarea întreruperilor în sistemele TR stricte reprezintă o problemă din punctul de vedere al predictibilității, afectând capacitatea acestora de a garanta în orice condiții de operare respectarea termenelor impuse task-urilor TR stricte.

În consecință, pentru a oferi predictibilitate maximă sistemelor TR stricte, este necesară abordarea modelelor non-preemptive în ceea ce privește deținerea semnalelor, a task-urilor, conceperea algoritmilor de planificare și implementarea sistemului.

Nucleul de operare TR HARETICK, prezentat în lucrarea de față, este conceput și implementat pe baza principiilor de mai sus. HARETICK pune la dispoziția aplicațiilor TR o serie de structuri și mecanisme, menite să maximizeze predictibilitatea operării și să garanteze respectarea termenelor stricte impuse task-urilor în faza de proiectare/specificare:

- ModX-urile sunt modele ale task-urilor TR stricte, periodice, modulare;
- Planificarea ModX-urilor se efectuează într-o manieră complet predictibilă, aplicând algoritmi non-preemptivi, special concepuți;
- Execuția ModX-urilor are loc într-un context strict, conform planificării, fiind controlată de către un executiv TR al nucleului;
- Operarea executivului TR (HDIS) se bazează pe întreruperea de ceas sistem (RTC) – singura întrerupere existentă în sistem.

Contextul de execuție descris mai sus este denumit *contextul strict* al nucleului HARETICK (sau, *contextul HRT* – Hard Real-Time), și reprezintă modalitatea de bază sub care nucleul tratează operarea task-urilor TR stricte ale unei aplicații.

Pe de altă parte, o serie de considerente au determinat introducerea și implementarea unui al doilea context de execuție în cadrul sistemului HARETICK:

- Majoritatea aplicațiilor TR, indiferent de tipul sau complexitatea acestora, conțin, pe lângă task-urile TR stricte, și task-uri cu specificații lejere de comportare în timp, sau chiar task-uri pentru care coordonata temporală nu e deloc importantă.

- Planificarea și execuția în context strict, non-preemptiv a ModX-urilor este foarte puțin flexibilă și implică o eficiență scăzută de operare a sistemului.
- Planificarea și execuția ModX-urilor se bazează pe timpii maximi de execuție a acestora, care se determină într-o manieră pesimistă (calea de execuție din secvența de instrucțiuni a ModX-ului, cu durata cea mai mare).
- Existența ModX-urilor fantomă (vezi capitolul anterior).

Pe baza acestor considerente, a fost introdus un context de execuție complementar cu cel strict, anume *contextul lejer* al nucleului HARETICK (sau, *contextul SRT* – Soft Real-Time). În cadrul contextului lejer, sunt planificate și executate task-urile sistem și ale aplicației, care nu impun cerințe stricte de comportare temporală, și pentru care garantarea respectării termenelor nu reprezintă o necesitate.

Task-urile lejere, notate cu L_i (spre deosebire de ModX-uri, care sunt notate cu M_i), se execută în regim preemptiv, în intervalele de timp rămase libere din cadrul contextului strict. Planificarea task-urilor lejere este efectuată de către un task sistem, SSCD (Soft real-time Scheduler – la rândul lui, un task lejer), pe baza unor algoritmi ce utilizează priorități și mecanisme de tip *time-sharing*.

Observație

Contextul strict de execuție poate fi privit ca o "întrerupere prelungită" a celui lejer, conform unei planificări riguroase, stricte, analizată apriori din punctul de vedere al fezabilității (analiză "offline").

Un alt mod de a vedea problema este acela că, deasupra contextului lejer (reprezentând modul de operare tradițional, preemptiv, al sistemelor de calcul) se poate implementa, ca o facilitare de execuție cu garantarea respectării termenelor stricte și cu predictibilitate maximă, contextul strict, a cărui întrerupere (RTC) este cea mai prioritară în sistem. În plus, pe durata execuției în contextul strict, celelalte întreruperi sunt suspendate.

Componenta sistem care asigură comutarea operării între cele două contexte de execuție, este executivul TR al HARETICK, HDIS. Principiile operațiilor de comutare între contextul strict și cel lejer, aplicate de HDIS, sunt:

- În momentul apariției unei întreruperi RTC, executivul activează contextul strict de execuție, lansând ModX-ul planificat la acel moment;
- După terminarea execuției unui ModX (sau în cazul unui ModX fantomă), HDIS verifică intervalul de timp rămas disponibil până când este planificat următorul ModX, și în caz afirmativ, restaurează contextul de execuție al ultimului task lejer întrerupt de sistem.

O exemplificare detaliată a modului de operare al HARETICK, cu cele două contexte de execuție, este prezentată în Figura 6-1. Utilizând aceeași axă a timpului, în figură sunt reprezentate graficele de operare ale contextului strict (HRT) și lejer (SRT), din punctul de vedere al planificării (partea superioară) și al execuției efective în sistem (partea inferioară).

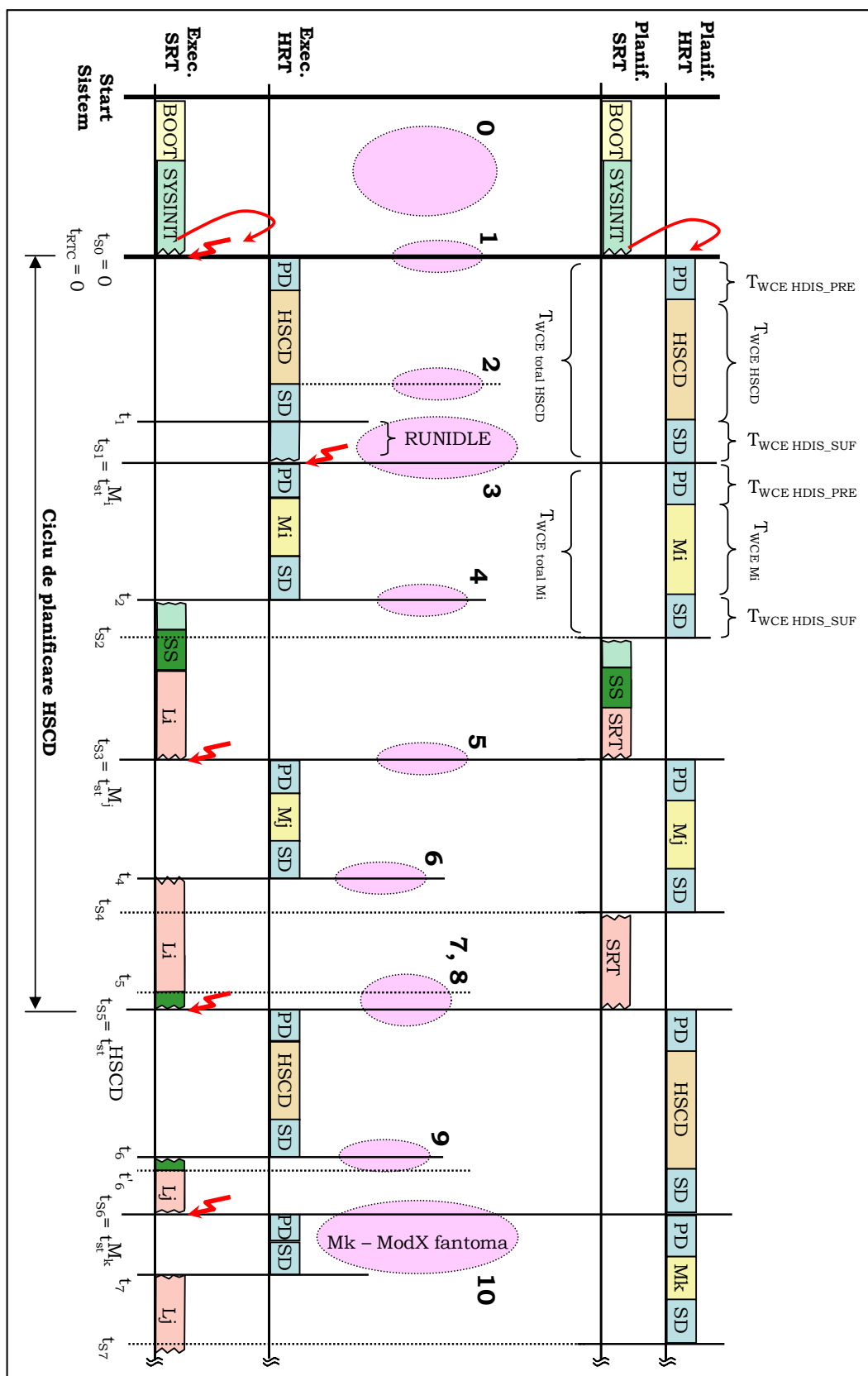


Figura 6-1. Prezentare detaliată a execuției task-urilor în HARETICK

Notățiile utilizate în Figura 6-1 pentru componentele HARETICK sunt după cum urmează:

- "PD" – componenta prefix a executivului TR, $_{HDIS_PRE}$;
- "SD" – componenta sufix a executivului TR, $_{HDIS_SUF}$;
- "SS" – planificatorul task-urilor contextului lejer de execuție, $_{SSCD}$;
- "Li" – task-ul lejer i al sistemului/aplicației;
- "Mi" – ModX-ul i al sistemului/aplicației;
- " t_{sk} " – momentul k al planificării (scheduling time), echivalent cu momentul de start sau de terminare a planificării unui ModX.

Pe parcursul operării nucleului HARETICK, se remarcă următoarele evenimente (situații) speciale, marcate în ordinea corespunzătoare pe grafurile din Figura 6-1:

- 0) Inițial, după startarea sistemului, rulează secvența BOOT care, după încărcarea celorlalte componente de bază ale HARETICK în memorie, activează task-ul lejer SYSINIT.

SYSINIT efectuează inițializările necesare, după care setează ceasul timp-real (RTC) al sistemului pe 0 ($t_0 = 0$), registrele de comparare ale timerelor RTC, de asemenea pe 0, și activează întreruperea RTC.

- 1) În momentul activării întreruperilor RTC, task-ul SYSINIT este întrerupt și se lansează (se comută) contextul strict de execuție al HARETICK, prin apelarea componentei prefix a executivului, $_{HDIS_PRE}$.

La apelarea $_{HDIS_PRE}$ din rutina de tratare a întreruperii RTC, acesta preia din Tabela de Dispatch (de la poziția curentă a pointerului) informația legată de execuția singurului ModX prezent deocamdată în tabelă: planificatorul contextului strict de execuție, HSCD.

Odată lansat în execuție, HSCD realizează planificarea non-preemptivă a setului de ModX-uri din sistem. Ciclul de planificare este calculat până ce HSCD completează toate înregistrările normale din Tabela de Dispatch, cu informația de planificare a ModX-urilor din sistem.

Pentru cazul considerat în exemplul nostru, Tabela de Dispatch conține o înregistrare specială (corespunzând lui HSCD) și două înregistrări normale, destinate planificării ModX-urilor din sistem pentru un ciclu de planificare. După prima execuție a lui HSCD, planificarea contextului strict prevede ca la momentul terminării lui HSCD (momentul t_{S_1}), va fi lansat în execuție ModX-ul M_i , a cărui execuție va dura până la momentul t_{S_2} . Mai departe, la momentul t_{S_3} , va fi executat ModX-ul M_j , care se termină la t_{S_4} . M_j încheie primul ciclu de planificare, iar conținutul înregistrării speciale corespunzând lui HSCD din tabelă va fi înlocuit cu instanța t_{S_5} – momentul de start al execuției următoare a HSCD. Execuția lui HSCD la t_{S_5} va iniția un alt ciclu de planificare (aici – al doilea).

Figura 6-2 ilustrează modul în care HSCD modifică conținutul Tabelei de Dispatch, de la starea inițială (setată de SYSINIT), la starea corespunzătoare terminării algoritmului de planificare pentru ciclul curent.

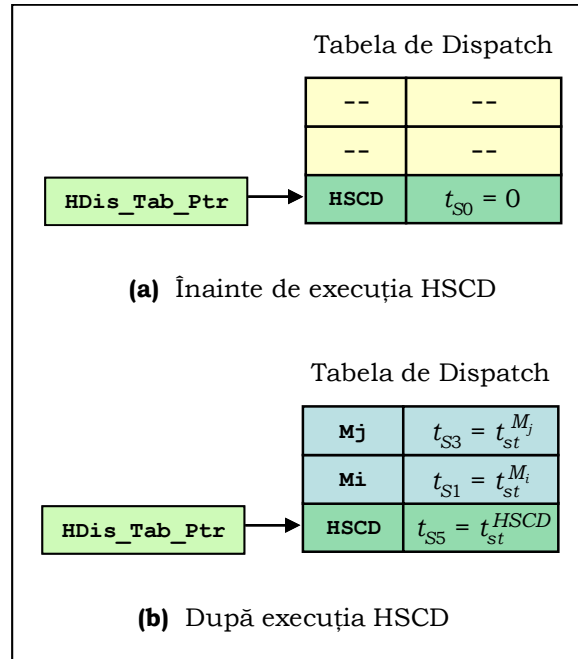


Figura 6-2. Conținutul Tabelei de Dispatch înainte și după execuția lui HSCD

Observație

Algoritmii de planificare non-preemptivă considerați pentru implementarea lui HSCD operează pe baza timpului maxim de execuție al ModX-urilor (WCET). Așa cum se vede și în Figura 6-1, durata maximă de execuție a oricărui ModX include și WCET al executivului TR al sistemului, HDIS (cu cele două componente ale sale – prefix și sufix).

Aceasta este o altă cerință importantă impusă de implementarea nucleului HARETICK asupra calculului timpilor WCET ai ModX-urilor.

- 2) Terminarea execuției ModX-ului HSCD se face cu apelarea componentei sufix a executivului TR al HARETICK, `HDIS_SUF`, care verifică în primul rând dacă a fost apelat de un ModX normal sau de către HSCD. În situația de față, `HDIS_SUF` preia din Tabela de Dispatch momentul de start al ModX-ului ce urmează în cadrul planificării, după HSCD (ModX-ul M_i), și programează regiștrii de comparare ai RTC cu valoarea respectivă ($t_{st}^{M_i}$), astfel încât, la momentul $t_{st}^{M_i}$, RTC va genera o nouă întrerupere în sistem.

În continuare, `HDIS_SUF` avansează pointerul în Tabela de Dispatch (în acest punct, pointerul va indica spre ModX-ul M_i), și determină intervalul de timp rămas disponibil (din momentul curent) până la momentul planificării ModX-ului M_i .

În Figura 6-1 se observă că execuția HSCD se termină mai repede (t_1), față de momentul considerat de planificare (t_{s1}).

- 3) Situația reprezentată de evenimentul (3) în Figura 6-1, este singurul caz în care executivul HDIS însuși este întrerupt de către RTC. Este vorba de starea specială a lui HDIS_SUF , denumită "RunIdle", și care reprezintă practic o buclă de așteptare a întreruperii, implementată de HDIS_SUF .

În faza de calculare a timpului disponibil (din momentul curent) până la momentul planificării lui M_i , HDIS_SUF se bazează pe o evaluare ce ține cont de o serie de parametri de eficiență a operării sistemului. Se apreciază astfel, dacă timpii consumați de restaurarea și salvarea contextului (a stării procesor) pentru task-ul lejer întrerupt, sunt suficient de mici comparativ cu timpul rămas pentru continuarea execuției acestuia, pentru ca operația de comutare a contextelor de execuție să fie eficientă.

La momentul t_1 , această evaluare determină decizia intrării lui HDIS_SUF în starea "RunIdle", adică de a nu comuta contextul strict de execuție. Se va aștepta astfel, apariția întreruperii RTC, care declanșează execuția următorului ModX (M_i).

- 4) La momentul t_2 , se termină execuția lui HDIS_SUF corespunzătoare ModX-ului M_i . Intervalul de timp de la t_2 , la $t_{S_3} = t_{st}^{M_j}$ (momentul planificării lui M_j), este suficient de mare pentru a permite comutarea contextului strict în contextul lejer de execuție.

Principala operație realizată la activarea contextului SRT de către HDIS este restaurarea contextului de execuție al ultimului task lejer întrerupt în sistem. În cazul de față, este vorba de task-ul SYSINIT, care își va continua execuția cu inițializarea celorlalte task-uri lejere din sistem, după care va lansa planificatorul contextului SRT de operare al HARETICK. SSCD va lansa, la rândul lui, în execuție task-ul lejer L_i .

- 5) Contextul lejer de execuție (task-ul L_i) este întrerupt de către RTC, la momentul planificării ModX-ului M_j , $t_{S_3} = t_{st}^{M_j}$. Rutina de tratare a întreruperii apelează executivul TR (HDIS_PRE), care salvează prima dată contextul de execuție al task-ului lejer L_i .
- 6) Componenta sufix a lui HDIS este apelată la terminarea ModX-ului M_j , când se va efectua comutarea contextelor. HDIS_SUF restaurează contextul de execuție a task-ului lejer întrerupt ultima dată (L_i), iar acesta își va continua execuția.
- 7, 8) Întreruperea RTC pentru lansarea în execuție a următorului ModX (aici, HSCD) apare în timpul execuției planificatorului task-urilor lejere, SSCD. HDIS va proceda la salvarea contextului de execuție al SSCD, la fel ca și cu celelalte task-uri lejere din sistem.
- 9) Restaurarea contextului de execuție al lui SSCD are loc la momentul t_6 , după care acesta își continuă operarea, lansând în execuție un alt task lejer, L_j .

- 10) Cazul în care executivul HDIS constată că ModX-ul care urmează a fi lansat în execuție (conform planificării) este un ModX fantomă.

Înainte de lansarea în execuție a unui ModX, `HDIS_PRE` verifică valoarea contorului de execuție al ModX-ului (`Mk.ECT`), din tabela descriptorilor de procese (PDT). Dacă `Mk.ECT = 0`, `HDIS_PRE` o lasă nemodificată, și în locul lansării în execuție a lui M_k , va apela componenta sufix, `HDIS_SUF`.

6.2 Stările ModX-urilor

Operarea ModX-urilor în cadrul sistemului HARETICK poate fi descrisă printr-un set de cinci stări:

- (1) Starea NOP ("No Operation").

Starea NOP descrie ModX-urile care nu sunt încărcate în cadrul nucleului HARETICK la un moment dat, ci se află în stadiu de dezvoltare și analiză în cadrul mediului integrat INVERTA, pe un calculator gazdă. Sistemul HARETICK nu are practic cunoștință despre aceste ModX-uri, ele neinfluențând în nici un fel operarea curentă a nucleului de pe platforma țintă.

Ca observație, remarcăm faptul că ModX-urile aparținând nucleului HARETICK (de exemplu, HSCD, HDIS, componentele stricte ale lui DATALINK, etc.) nu se vor găsi niciodată în starea NOP, ele fiind încărcate pe platforma țintă odată cu restul nucleului HARETICK în timpul operației de BOOT. Doar ModX-urile aplicațiilor ce vor fi încărcate în sistem la un moment dat, pot să se găsească în starea NOP, înaintea încărcării acestora.

- (2) Starea RDY ("Ready").

Odată ce o aplicație oarecare a fost încărcată pe platforma țintă pe care rulează nucleul HARETICK, ModX-urile acesteia vor intra în starea RDY ("gata de planificare").

Un ModX în starea RDY este identificat și reprezentat complet în cadrul nucleului, având alocate majoritatea resurselor necesare execuției:

- identificare în cadrul aplicației de care aparține, prin tabela PDAGT;
- caracterizare completă prin parametrii săi din tabela PDT;
- alocarea zonei proprii de cod în memoria program a sistemului;
- alocarea zonei parametrilor de ieșire ("Data Out") în memoria de date;
- alocarea zonei cu variabilele globale ale aplicației și identificarea celor utilizate de ModX;
- crearea tabelii de simboluri a aplicației.

(3) Starea "SCD" ("Scheduled").

ModX-urile din starea SCD aparțin planificării din cadrul ciclului curent, ca rezultat al aplicării algoritmilor de planificare strictă, non-preemptivă, implementați de HSCD, asupra setului de ModX-uri sistem și ale aplicației.

ModX-urile din starea SCD se regăsesc în cadrul Tabelei de Dispatch, rezultând că la un moment dat (într-un ciclu de planificare), există un număr bine determinat de ModX-uri în starea SCD. Numărul acestora depinde de lungimea tabelului, mai puțin prima înregistrare, care referă de fiecare dată pe HSCD.

(4) Starea "RUN" ("În execuție").

În starea RUN se poate găsi, la un moment dat, un singur ModX. Acestuia i se alocă restul resurselor sistem necesare execuției efective pe platforma țintă:

- zona "Heap" a sistemului, alocată exclusiv pentru parametrii de intrare și variabilele locale ale ModX-ului, pe toată durata execuției sale, și numai atunci;
- procesorul sistemului, care execută în regim non-preemptiv secvența de instrucțiuni a ModX-ului.

(5) Starea "GST" ("Ghost").

Starea GST este starea în care se află la un moment dat un ModX fantomă. ModX-ul fantomă este caracterizat prin faptul că, deși a fost planificat de către HSCD, în momentul lansării în execuție are contorul de execuții nul.

ModX-urile fantomă nu sunt lansate în execuție efectivă de către nucleu (adică de către executivul TR al nucleului, HDIS), deci nu li se alocă resursele suplimentare necesare execuției, ca în cazul stării RUN.

Trecerea unui ModX oarecare dintr-o stare în alta are loc în urma îndeplinirii unor condiții specifice, și are loc prin intermediul unui set distinct de operații. Figura 6-3 ilustrează cele cinci stări posibile, împreună cu configurația resurselor sistemului pentru fiecare stare în parte. Figura prezintă de asemenea, și modurile posibile de trecere a ModX-urilor dintr-o stare în alta, precum și task-urile sistem implicate în aceste operații.

Încărcarea unei aplicații oarecare în sistem este efectuată de către task-ul LOADER, prin intermediul componentei sistem DATALINK. LOADER-ul este responsabil de alocarea și completarea structurilor necesare reprezentării aplicației și a ModX-urilor acesteia în cadrul nucleului HARETICK: tabela PDAGT, tabela PDT, tabela de simboluri, zonele de cod și "Data Out" pentru fiecare ModX, zona variabilelor globale, etc.

Setul de operații ce corespunde trecerii ModX-urilor din starea NOP în RDY este denumit "load".

Descărcarea (eliminarea) unei aplicații din sistemul HARETICK este realizată tot de task-ul LOADER, eliberându-se resursele sistem utilizate de nucleu la reprezentarea și gestionarea aplicației. Setul de operații corespunzător trecerii unui ModX din starea RDY în starea NOP este denumit "unload".

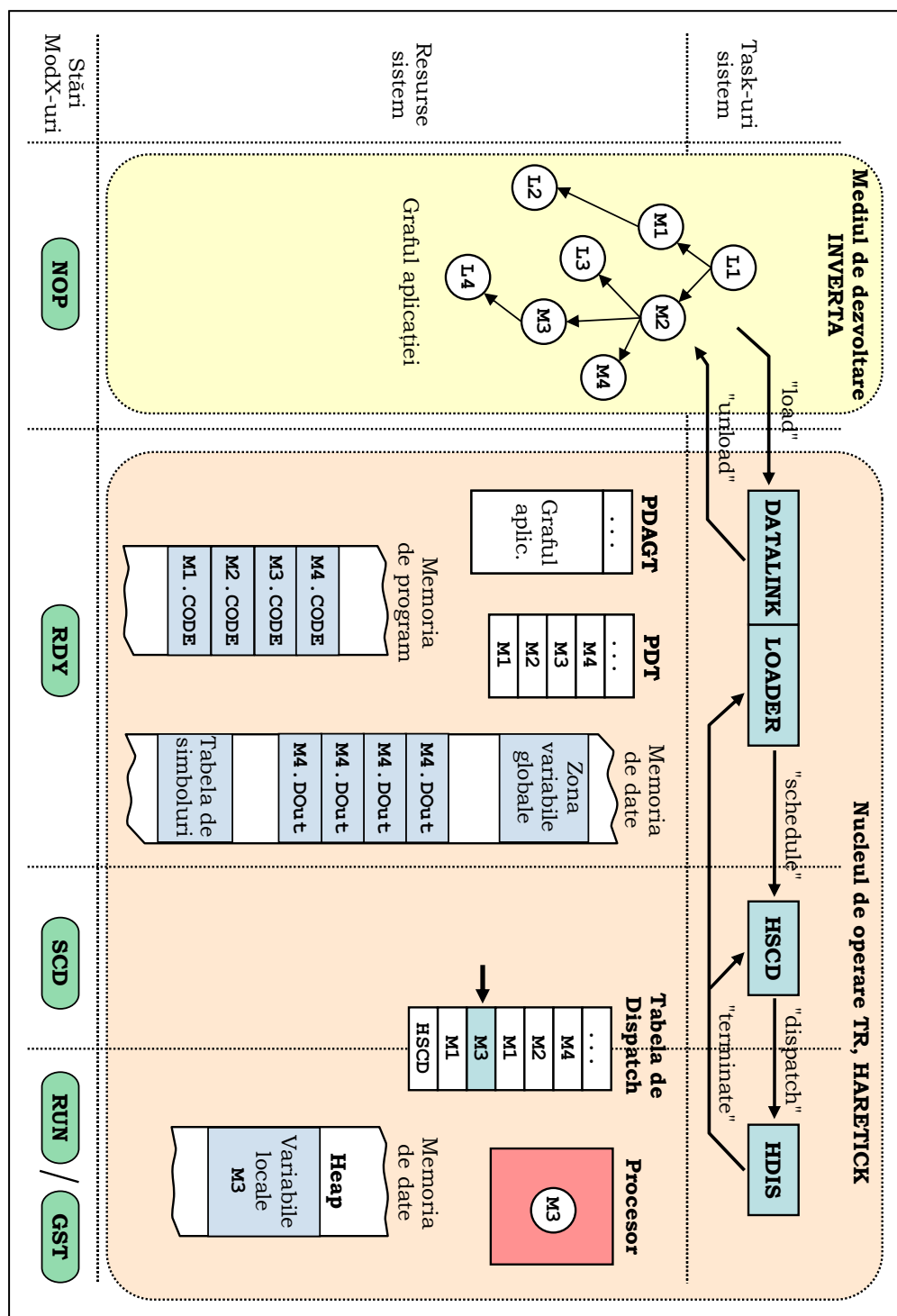


Figura 6-3. Stările ModX-urilor și operarea nucelului HARETICK

Planificatorul contextului strict de execuție, HSCD, aplică algoritmul de planificare non-preemptivă special conceput pentru sistemul HARETICK, asupra setului de ModX-uri aflate în starea RDY. În cadrul unui ciclu de planificare, va fi selectat un număr bine definit de ModX-uri, cu care se construiește (completează)

Tabela de Dispatch. Ca urmare, starea ModX-urilor planificate se schimbă, din RDY în SCD.

Ca observație, menționăm că un anumit ModX poate apărea în cadrul mai multor înregistrări din Tabela de Dispatch, fiind astfel planificat pentru execuție de mai multe ori în ciclul curent de planificare. De exemplu, un ModX cu perioadă mai mică are o frecvență mai mare de apariții în cadrul Tabelei de Dispatch, comparativ cu un ModX cu perioadă mai mare.

Setul de operații prin care un ModX trece din starea RDY în SCD este denumit "schedule". Cum se observă și din Figura 6-3, arhitectura nucleului HARETICK nu permite tranziția inversă a ModX-urilor, anume din SCD, direct în RDY. Cu alte cuvinte, odată ce un ModX a fost planificat pentru ciclul curent de planificare, el va fi preluat în continuare de executivul TR al nucleului, HDIS, în vederea lansării în execuție la momentul specificat în planificare.

ModX-urile planificate de către HSCD sunt preluate pe rând de către executivul HDIS, prin baleierea Tabelei de Dispatch, în vederea lansării în execuție pe baza mecanismului de întreruperi RTC. La un moment dat, ceasul timp-real al sistemului va genera o întrerupere, semnalând instanța de timp pentru care a fost planificată execuția ModX-ului curent referit în Tabela de Dispatch. Rutina de tratare a întreruperilor apelează executivul HDIS, care verifică valoarea contorului de execuții a ModX-ului curent. În cazul în care contorul de execuții este nul, ModX-ul devine un ModX fantomă, trecând astfel în starea GST. HDIS nu îl va lansa în execuție efectivă, ci va trece la următorul ModX din tabelă.

Prin urmare, ModX-urile din starea GST sunt considerate de sistem ca "fiind executate", fără a fi lansate efectiv în execuție (de aici și denumirea de "ModX fantomă").

În cazul în care contorul de execuții al ModX-ului curent ce urmează a fi executat este nenul, executivul HDIS îi alocă restul resurselor sistem necesare execuției acestuia (zona "Heap" pentru parametrii de intrare și variabilele locale, și procesorul sistemului). ModX-ul trece astfel în starea RUN, ocupând exclusiv procesorul sistemului pentru execuția în regim non-preemptiv a secvenței proprii de instrucțiuni.

Setul de operații implicate în tranziția ModX-urilor din starea SCD în RUN sau GST, este denumit "dispatch".

După execuția (efectivă sau "fantomă") a unui ModX aflat în starea RUN, respectiv GST, acesta poate reveni în starea RDY sau SCD. În primul caz, ModX-ul care tocmai a fost executat și-a terminat și planificările din ciclul curent de planificare, adică nu mai există o altă înregistrare corespunzătoare aceluiași ModX în Tabela de Dispatch. Cea de-a doua variantă, presupune contrariul celor de mai sus, adică ModX-ul respectiv se mai regăsește în cadrul tabelii, urmând a mai fi executat în același ciclu de planificare. El rămâne astfel, în starea de ModX planificat (SCD).

Setul operațiilor implicate de terminarea execuției ModX-urilor și trecerea lor din starea {RUN, GST} în starea {SCD, RDY}, este denumit "terminate".

Tabelul 6-1 oferă o sinteză a celor discutate până acum în legătură cu stările ModX-urilor sistemului HARETICK și tranzițiile posibile între stări.

Tabelul 6-1. Sinteză a stărilor ModX-urilor și a tranzițiilor între stări

Tranziție stări	Denumire	Semnificație
NOP → RDY	"load"	Încărcarea unei aplicații dezvoltate și analizate în mediul integrat INVERTA de pe un calculator gazdă, pe platforma țintă ce rulează sub nucleul TR de operare HARETICK.
RDY → NOP	"unload"	Descărcarea (eliminarea) unei aplicații din sistemul HARETICK și eliberarea resurselor sistem alocate pentru reprezentarea și gestionarea ei.
RDY → SCD	"schedule"	Includerea unui subset de ModX-uri ale sistemului (aplicației) în ciclul curent de planificare, în vederea execuției pe sistemul HARETICK.
SCD → {RUN, GST}	"dispatch"	Preluarea de către executivul HDIS a unui ModX inclus în ciclul curent de planificare, în vederea execuției. În cazul în care contorul său de execuții este nenul, ModX-ul va fi executat efectiv (starea RUN), altfel, se va considera doar că a fost executat (starea GST).
{RUN, GST} → {RDY, SCD}	"terminate"	Terminarea execuției (efective – RUN, sau fictive – GST) a unui ModX, urmată de revenirea acestuia în starea "gata de planificare" (RDY) sau "planificat și gata de execuție" (SCD), în cazul în care ciclul curent de planificare prevede mai multe execuții ale ModX-ului.

7 Executivul TR al HARETICK: HDIS

7.1 Caracteristici generale

Nucleul de operare timp-real HARETICK, propus și prezentat în lucrarea de față se bazează pe o serie de componente principale, printre care, executivul TR joacă un rol esențial.

Așa cum s-a văzut și în capitolele anterioare ale materialului (în special Capitolul 3 și 6), executivul TR al HARETICK, HDIS (Hard real-time Dispatcher), are principala sarcină de a lansa în execuție ModX-urile sistem și ale aplicației aflate în starea SCD ("planificat") și de a termina execuția acestora, în contextul strict de execuție al nucleului HARETICK. HDIS implementează astfel, mecanismele necesare comutării celor două tipuri de contexte de execuție existente în sistem: contextul strict (contextul HRT – Hard Real-Time) în cadrul căruia se execută în regim non-preemptiv ModX-urile sistemului, și contextul lejer (contextul SRT – Soft Real-Time) ce corespunde execuției în regim preemptiv a task-urilor lejere.

Operarea HDIS se face într-o manieră complet predictibilă, bazată pe următoarele considerente:

- HDIS este apelat de către rutina de tratare a întreruperii de ceas timp-real (RTC), la momente de timp bine stabilite, conform planificării execuției task-urilor TR stricte în sistem;
- Durata maximă de execuție a HDIS ($WCET_{HDIS}$) este calculată cu atenție, luând în considerare toate situațiile posibile de operare ale nucleului. Pe orice ramură de proiectare s-ar afla la un moment dat execuția codului HDIS, timpii maximi de execuție pot fi determinați cu precizie, fără a depinde de structuri sau mecanisme dinamice (cum ar fi de exemplu, numărul total de ModX-uri în sistem, numărul parametrilor de intrare sau de ieșire al acestora, etc.);
- Întreaga arhitectură a HARETICK și a mecanismelor de reprezentare și gestionare a ModX-urilor în cadrul nucleului, este concepută și proiectată în așa fel încât să faciliteze operarea predictibilă a executivului și a întregului sistem. Sunt utilizate doar reprezentări statice și operații precis limitate în timp și spațiu. În orice moment, sunt cunoscute cu precizie valorile maxime ale dimensiunilor structurilor folosite și duratele maxime de execuție ale operațiilor.

Executivul HDIS are două componente principale: un prefix (HDIS_PRE) și un sufix (HDIS_SUF), cu roluri bine determinate. Astfel, HDIS_PRE inițiază execuția ModX-ului planificat a fi lansat la un moment dat, și este apelat de întreruperea RTC

ce apare la acel moment în sistem. Odată activat, HDIS_PRE efectuează următoarele acțiuni:

- verifică dacă a fost întrerupt contextul lejer. În caz afirmativ, se efectuează salvarea contextului de execuție lejeră a task-ului întrerupt într-o zonă dedicată;
- identifică ModX-ul ce urmează a fi lansat în execuție la momentul curent, din Tabela de Dispatch;
- citește următoarea înregistrare din Tabela de Dispatch, corespunzând ModX-ului ce urmează ca planificare celui curent, și extrage momentul de start a execuției acestuia;
- programează timerele ceasului de timp-real (RTC) pentru a genera întreruperi la momentul de start a ModX-ului următor;
- decrementează contorul de execuții a ModX-ului curent (dacă acesta are valoare finită);
- trece ModX-ul curent din starea "planificat" ("SCD"), fie în starea "de execuție" ("RUN") – dacă valoarea contorului de execuție era nenulă înaintea decrementării, fie în starea "ModX fantomă" ("GST");
- dacă ModX-ul curent este în starea "RUN", îi cedează procesorul și îl apelează pentru execuție.

Pe de altă parte, componenta sufix a lui HDIS este apelată la terminarea execuției fiecărui ModX, chiar de către acesta. Odată activat, HDIS_SUF efectuează următoarele acțiuni:

- trece ModX-ul curent din starea "de execuție" (RUN) sau "ModX fantomă" (GST), înapoi în starea "planificat și gata de execuție" (SCD) sau în "gata de planificare" (RDY);
- avansează referința (pointerul) de citire al Tabelei de Dispatch către ModX-ul următor;
- verifică dacă există suficient timp pentru a comuta contextul strict cu cel lejer, permițând execuția task-urilor lejere ale sistemului și aplicației. Dacă nu este timp suficient, HDIS va iniția o buclă infinită ("RunIdle") în care va aștepta apariția întreruperii RTC ce marchează evenimentul lansării în execuție a ModX-ului următor. Dacă se decide comutarea controlului către contextul lejer, are loc restaurarea stării de execuție (a contextului) task-ului lejer care a fost întrerupt.

Pe lângă cele două componente principale, executivul HDIS mai utilizează o subrutină denumită "TiLT" ("Time Log Tool"), pentru generarea unor scurte rapoarte legate de evenimentele lansării în execuție și terminării ModX-urilor. Evenimentele sunt înscrise într-o tabelă dedicată, împreună cu valorile instanțelor de timp corespunzătoare.

7.2 Structurile de date utilizate de HDIS: Tabela de Dispatch

Structura de date de bază a executivului HDIS este Tabela de Dispatch (vezi și Capitolul 5 și 6). Tabela de Dispatch (Figura 7-1) este o structură sistem destinată gestionării unui număr constant de planificări și execuții de ModX-uri în cadrul contextului strict al nucleului HARETICK. Tabela are un număr fix, bine stabilit de înregistrări ($\lambda + 1$, conform figurii), din care prima este o înregistrare specială ce identifică execuția planificatorului contextului strict de execuție, HSCD, iar restul λ sunt înregistrări normale, identificând execuții de ModX-uri.

Fiecare înregistrare a Tabelei de Dispatch conține două câmpuri: (i) identificatorul ModX-ului, PID, și (ii) momentul de start al execuției ModX-ului, conform planificării realizate de HSCD în cadrul ciclului curent de planificare. Tabela este definită prin adresa de bază (`Sys_HDis_Table`) și de dimensiunea sa totală ($\lambda \cdot \langle \text{dimensiunea unei înregistrări} \rangle$).

Înregistrarea curentă în cadrul tabelii este referită printr-un pointer, `HDis_Tab_Ptr`. Adresarea tabelii cu ajutorul pointerului se face în mod circular, astfel încât o incrementare a pointerului în situația în care acesta indică spre ultima înregistrare a tabelii, va avea ca rezultat saltul pointerului spre prima locație din tabelă.

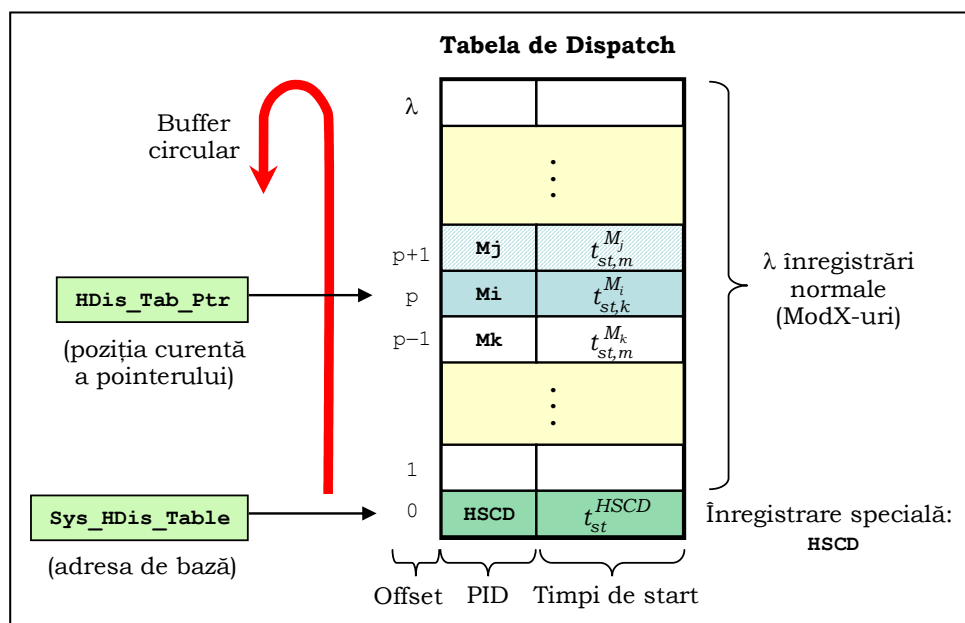


Figura 7-1. Structura Tabelei de Dispatch

Figura 7-2 ilustrează modul curent de implementare a Tabelei de Dispatch, pe arhitectura Motorola DSP56307. Fiecare înregistrare conține un cuvânt procesor pentru identificarea ModX-ului (PID) și două cuvinte (2·24 biți) pentru exprimarea timpului de start al execuției ModX-ului (`STime_Hi:STime_Lo`). Structura de timp a înregistrărilor tabelii este compatibilă cu arhitectura ceasului timp-real (RTC) care e

compusă din două timere cascade (vezi Capitolul 4 și Figura 4-3). În acest fel se ușurează operațiile de programare a regiștrilor de comparare ai RTC de către HDIS.

Dimensiunea totală a tabelului este de 1000 (înregistrări normale) + 1 (înregistrare specială), rezultând un număr de 3003 cuvinte de memorie. Prin urmare, un ciclu de planificare va cuprinde exact 1001 execuții de fiecare dată, din care una corespunde planificatorului HSCD.

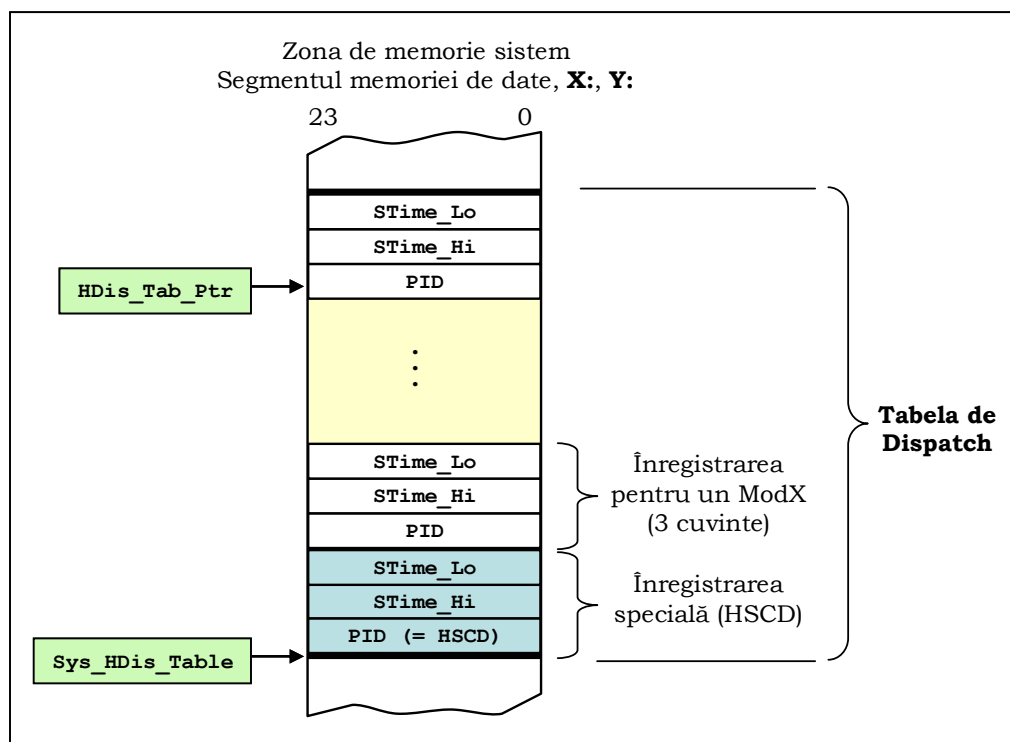


Figura 7-2. Implementarea curentă a Tabelei de Dispatch

Tabela de Dispatch este accesată în mod circular și doar în citire de către HDIS, de fiecare dată când este activat executivul ca urmare a unei întreruperi RTC. HDIS va identifica ModX-ul ce urmează a fi lansat în execuție la momentul apariției întreruperii și, de asemenea, va programa regiștrii de comparare ai RTC pentru a genera întreruperea corespunzătoare planificării ModX-ului următor.

Completarea înregistrărilor tabelului este efectuată de către planificatorul contextului strict de execuție al HARETICK, HSCD. La fiecare execuție a lui HSCD (corespunzând începutului unui nou ciclu de planificare), acesta va completa înregistrările normale ale tabelului (cele λ înregistrări, începând cu offset-ul 1) cu informația legată de planificarea ModX-urilor. La sfârșit, va modifica valoarea timpului de start pentru înregistrarea specială, cu momentul execuției HSCD ce inițiază ciclul următor de planificare.

7.3 Funcționarea de principiu

În continuare vom detalia pe rând funcționarea celor trei componente ale executivului HDIS: HDIS_PRE, HDIS_SUF și subrutina TiLT.

Separarea executivului în două componente, prefix și sufix, a fost realizată ținând cont de cele două ipostaze de apelare ale lui HDIS și de modul în care acesta încadrează execuția propriu-zisă a unui ModX. Componenta prefix este apelată de către întreruperea RTC și inițiază execuția ModX-urilor, pe când componenta sufix este apelată de către ModX-uri, la terminarea execuției acestora.

Ramurile de procesare ale executivului HDIS mai trebuie să trateze o situație importantă: execuția planificatorului contextului strict, HSCD. În cazurile în care ModX-ul curent ce urmează a fi lansat în execuție este identificat în Tabela de Dispatch ca fiind HSCD, executivul nu poate citi din tabelă timpul de start pentru ModX-ul următor, în vederea programării RTC. Acest lucru se datorează faptului că restul înregistrărilor din tabelă (ce identifică ModX-uri normale) urmează a fi completate chiar de către planificator, în timpul execuției sale. Prin urmare, în afara primei înregistrări, ce corespunde lui HSCD, Tabela de Dispatch poate fi considerată goală.

Soluția constă în faptul că, atunci când e vorba de lansarea în execuție a HSDC, programarea timerelor RTC nu se mai efectuează de către executivul prefix, ci de către componenta sufix, după terminarea execuției planificatorului ce are ca rezultat completarea restului tabelii.

Figura 7-3, continuată cu Figura 7-4, prezintă organigrama de operare a componentei prefix a executivului. HDIS_PRE este lansată în execuție de către întreruperea RTC, anume de întreruperea de `Timer0`, în implementarea curentă a nucleului HARETICK pe arhitectura Motorola DSP56307. Fiind vorba de două timere cascade ca bază de timp în sistem, `Timer0` și `Timer1`, a apărut problema rezolvării unei întreruperi suplimentare – cea generată de `Timer1`.

`Timer1` va genera întrerupere doar dacă între două planificări succesive de ModX-uri există un interval de timp ce depășește capacitatea curentă de numărare rămasă disponibilă în `Timer0` (registru de numărare al `Timer0` se va "da peste cap").

Problema a fost rezolvată, în primul rând prin verificarea apariției acestor situații de depășire a capacității primului timer. Verificarea se face comparând cuvintele cele mai semnificative ale valorilor timpilor de start, pentru ModX-ul curent și cel următor, din Tabela de Dispatch. Prin această comparare se verifică practic dacă valoarea din `Timer1` este aceeași pentru cei doi timpi, sau nu. Dacă este aceeași, înseamnă că timpul ce se va scurge de la execuția ModX-ului curent și până la lansarea următorului ModX, este suficient de mic pentru capacitatea de contorizare disponibilă în `Timer0`. Dacă se confirmă depășirea capacității curente a `Timer0`, se vor programa registrele de comparare ai ambelor timere, cu valorile corespunzătoare momentului de start al ModX-ului următor, și se va activa doar întreruperea pentru `Timer1`.

Rutina de tratare a întreruperii generate de `Timer1` are doar rolul de activare a întreruperii pentru `Timer0`. În arhitectura DSP56307, ea este denumită "întrerupere scurtă", pentru că nu implică salvarea stării procesor și a contorului de program în stivă.

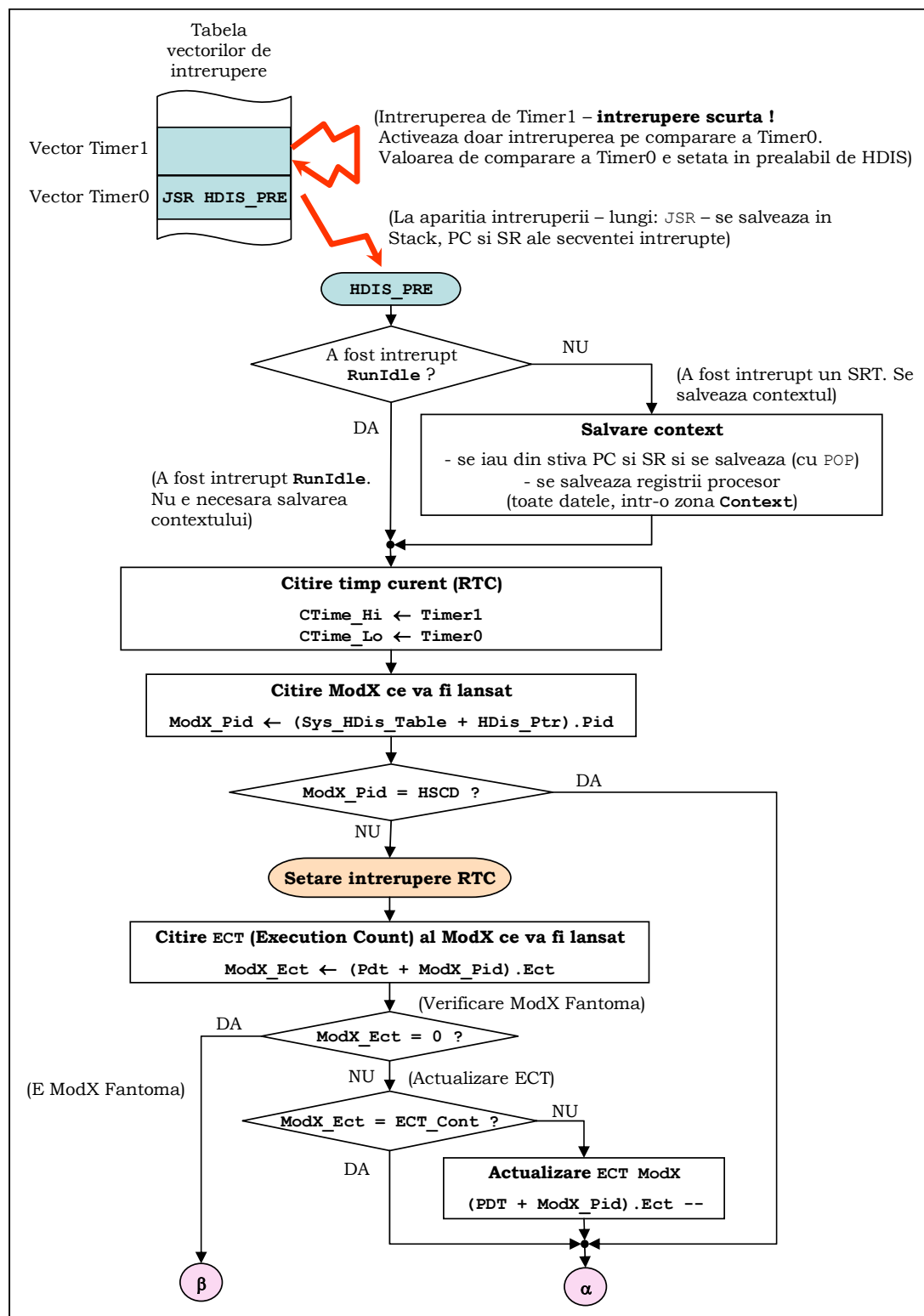


Figura 7-3. Organigrama de operare a componentei HDIS_PRE

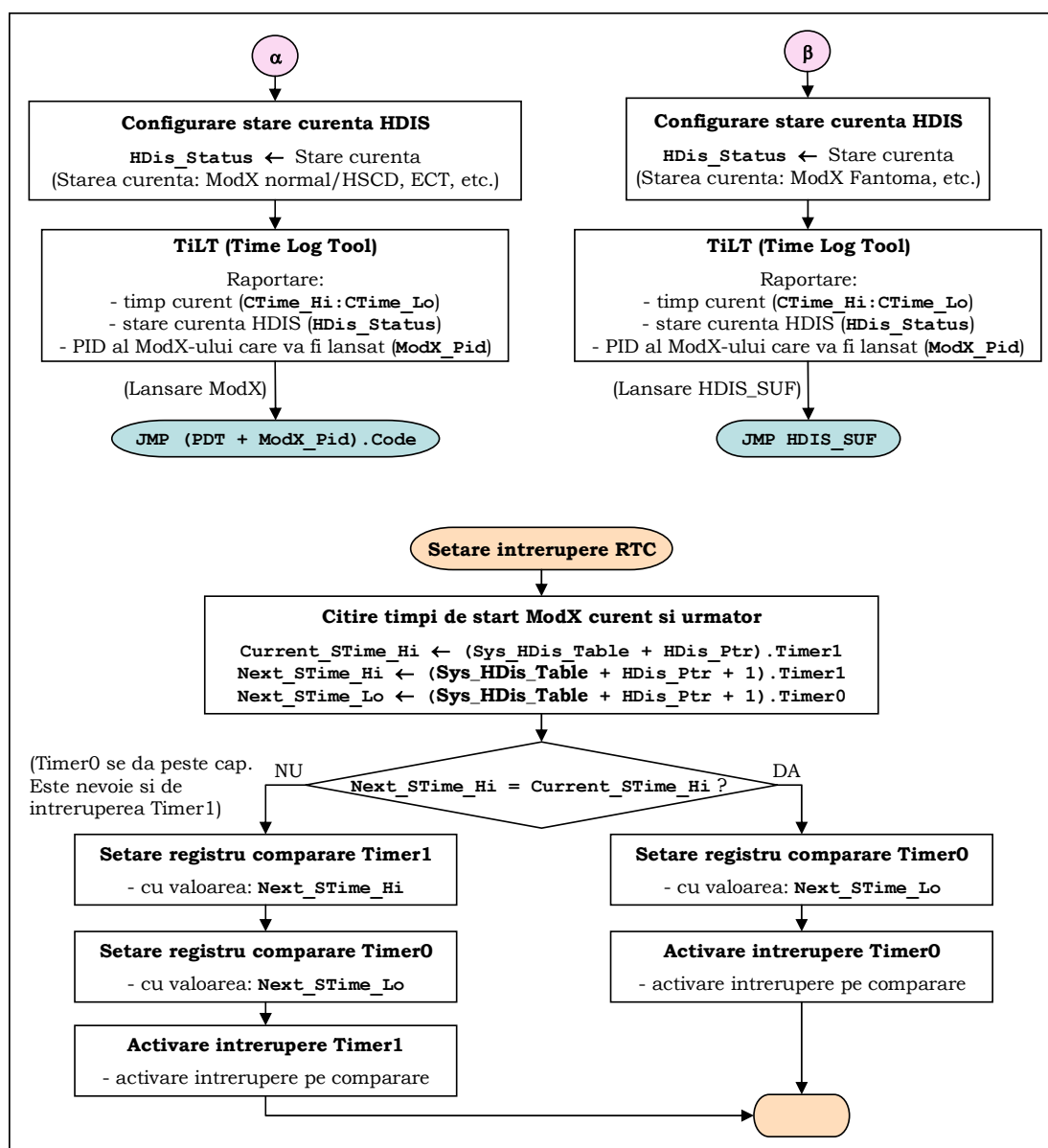


Figura 7-4. Organigrama HDIS_PRE (continuare)

După apelarea lui HDIS_PRE de către întreruperea generată de `Timer0`, se verifică prima dată dacă a fost întreruptă execuția unui task lejer, caz în care este necesară salvarea contextului său de execuție într-o zonă de memorie sistem dedicată, denumită "Context". Operația de verificare se bazează pe informația de stare pe care o menține HDIS de la o execuție la alta, prin intermediul unei variabile proprii (`HDis_Status`).

Figura 7-5, continuată cu Figura 7-6, prezintă organigrama de operare a componentei sufix a executivului. HDIS_SUF este apelată de către ModX-ul care își încheie execuția în contextul strict al nucleului.

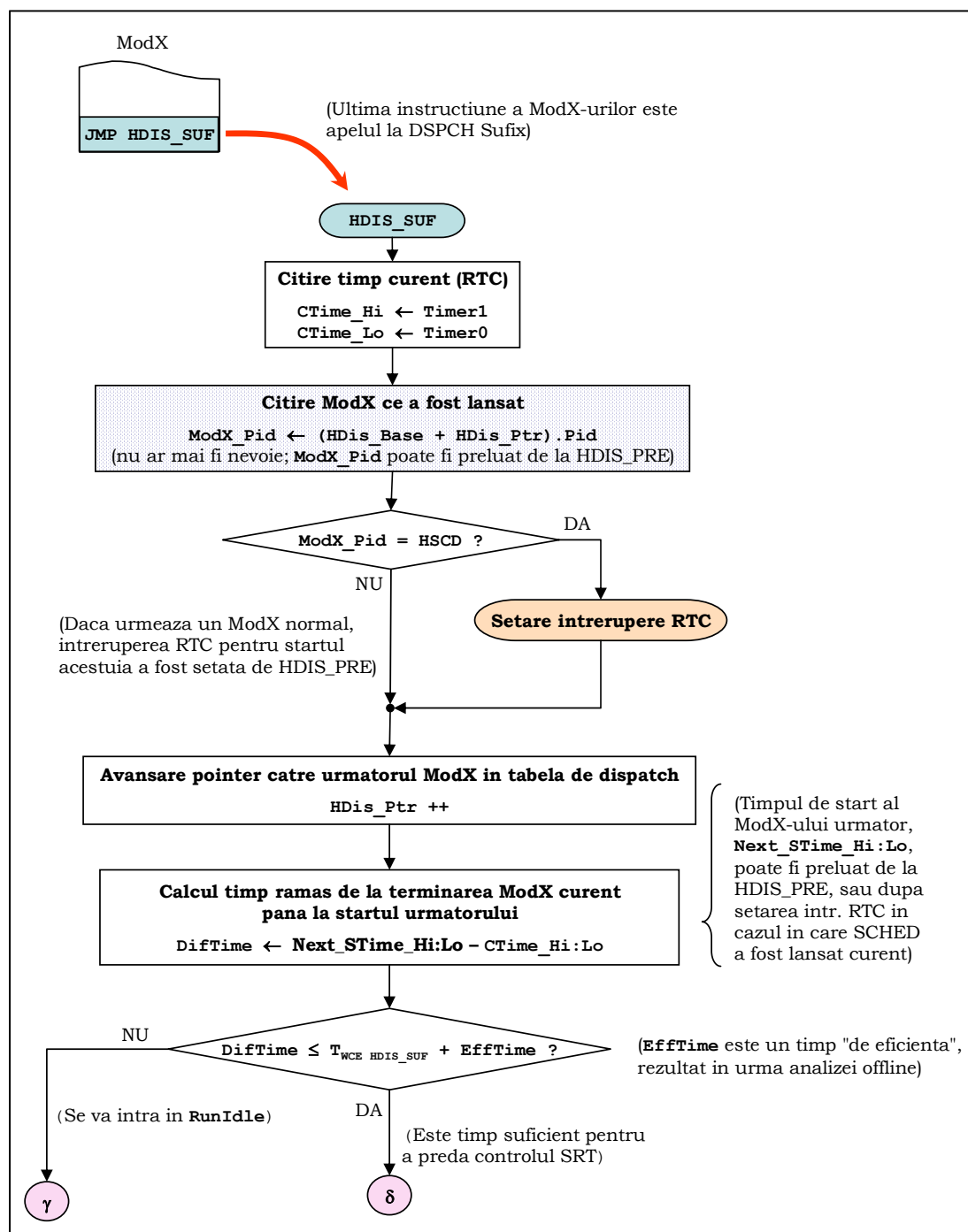


Figura 7-5. Organigrama de operare a componentei HDIS_SUF

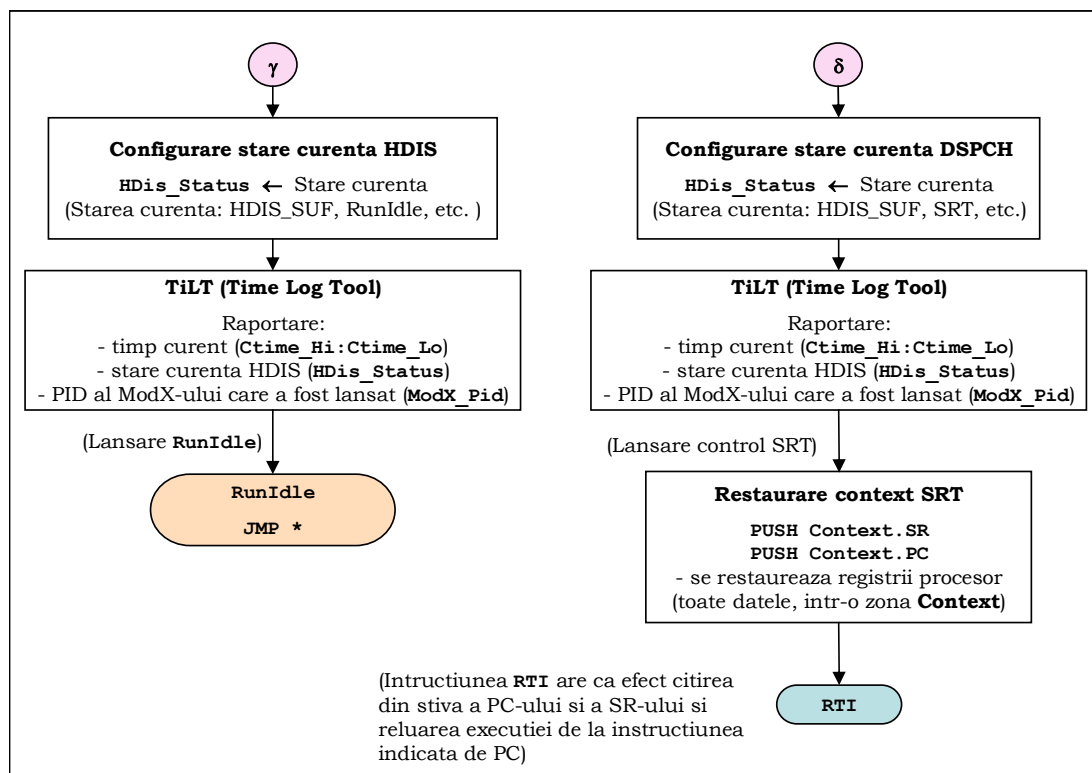


Figura 7-6. Organigrama HDIS_SUF (continuare)

Se poate observa din organigramele celor două componente ale HDIS că, de fiecare dată când sunt activate, are loc citirea și înregistrarea momentului curent de timp (se citesc contoarele de timer ale ceasului timp-real). Informația de timp este utilizată în rapoartele pe care HDIS le generează prin intermediul subrutinei TiLT.

Subrutina TiLT gestionează o tabelă (un buffer circular), în care înscrie, pe lângă momentul curent al apariției unui eveniment, informații legate de tipul evenimentului, starea curentă a HDIS și contorul de execuții al ModX-ului care a fost procesat de HDIS.

8 Concluzii și perspective

8.1 Concluzii

Referatul de față face parte din programul de doctorat cu tema "Contribuții la achiziția și prelucrarea numerică în timp-real a semnalelor utilizând sisteme de calcul distribuite", sub coordonarea științifică a prof. dr. ing. Vladimir CREȚU.

În cadrul lucrării a fost abordată problematica sistemelor timp-real stricte destinate aplicațiilor critice de achiziție și prelucrare numerică a semnalelor (DSP-based systems) și de control digital încorporat (embedded systems), din perspectiva proiectării și implementării acestora.

Lucrarea de față este o continuare a preocupărilor prezentate în referatul al doilea, în cadrul căruia a fost conceput și analizat un model complet, unitar, de sistem TR strict pentru aplicații critice.

Modelul propus se bazează pe câteva considerente principale care definesc liniile generale urmate în materialul de față, la proiectarea și implementarea acestuia:

- Utilizarea întreruperilor în sistemele TR stricte reprezintă o problemă din punctul de vedere al predictibilității, afectând capacitatea acestora de a garanta în orice condiții de operare respectarea termenelor impuse task-urilor TR stricte.
- Pentru a oferi predictibilitate maximă sistemelor TR stricte, este necesară abordarea modelelor non-preemptive în ceea ce privește definirea semnalelor, a task-urilor, conceperea algoritmilor de planificare și implementarea sistemului.
- În cadrul dezvoltării și implementării aplicațiilor TR, faza de analiză offline, apriori execuției pe sistemul țintă, a setului de task-uri TR stricte al aplicației, reprezintă o cerință obligatorie.
- Dezvoltarea viabilă a sistemelor și aplicațiilor TR stricte trebuie să aibă la bază, pentru toate fazele sale, modele și metode omogene, compatibile.

În consecință, modelul de sistem TR conceput cuprinde două elemente principale: un mediu integrat, destinat dezvoltării și analizei offline a aplicațiilor, și un nucleu de operare TR strict pe care se încarcă și se execută aplicațiile validate. OPEN-HARTS (Operanting Environment for Hard Real-Time Systems) este conceput ca un mediu de operare pentru sisteme și aplicații TR stricte, având două componente majore:

- (1) Mediul integrat de dezvoltare și analiză a aplicațiilor TR stricte pe sisteme DSP și de control digital încorporat, INVERTA (Integrated Visual Environment for Real-Time Applications Analysis and Development). INVERTA oferă

programatorului de sistem și de aplicații TR un mediu vizual integrat, destinat proiectării, specificării și verificării formale a aplicațiilor (cu accent pe comportamentul temporal al task-urilor – obligatoriu la cele TR stricte). INVERTA permite de asemenea, programarea funcțională a fiecărui task al aplicației, analiza temporală a task-urilor (extragerea WCET) și analiza planificabilității. Toate operațiile se realizează în regim "off-line" vis-a-vis de operarea sistemului TR propriu-zis, pe o platformă de calcul (Host platform), care poate fi statică (workstation), sau mobilă (Laptop, PDA, Tablet PC, etc.).

- (2) Nucleul de operare TR hibrid pentru sisteme încorporate, HARETICK (Hard Real-Time Compact Kernel). HARETICK este instalat pe platforma țintă de control digital încorporat, și permite încărcarea și execuția task-urilor TR stricte (Hard Real-Time tasks) în regim non-preemptiv și a celor lejere (Soft Real-Time tasks) în regim preemptiv.

În cadrul referatului de față sunt descrise principiile și metodele implicate în proiectarea și implementarea unui nucleu de operare TR strict, detaliind particularitățile întâlnite cu ajutorul unui studiu de caz: implementarea nucleului de operare HARETICK pe arhitectura Motorola DSP56307.

Caracteristicile generale ale nucleului HARETICK derivă din faptul că au fost vizate sistemele de control digital încorporat (embedded systems). Prin urmare, sistemul HARETICK prezintă următoarele caracteristici de bază:

- Este un nucleu de operare timp-real strict, oferind facilități pentru operarea cu predictibilitate maximă a aplicațiilor critice, cu garantarea respectării termenelor stricte de timp impuse de acestea.
- Este un nucleu de tip *single-user*, față de care, programatorul de sistem/aplicații poartă întreaga responsabilitate a proiectării, programării și utilizării sale.
- Este un nucleu *multitasking*, permițând execuția concurentă, planificată a unor seturi compuse din mai multe task-uri.
- Este conceput și proiectat într-o manieră puternic modularizată, vizându-se simplitatea și reducerea pe cât posibil a dimensiunilor nucleului.
- Timpul este un parametru esențial de operare al sistemului, fiind reprezentat și gestionat cu ajutorul unor structuri și mecanisme dedicate.
- Este un nucleu de operare hibrid, având două contexte de execuție complementare: contextul strict (sau contextul HRT) în care se execută task-urile TR stricte din sistem (ModX-urile) în regim non-preemptiv, și contextul lejer (sau contextul SRT), pentru execuția task-urilor lejere în regim preemptiv.

HARETICK este conceput ca un set omogen de componente software de bază, care interacționează strâns unul cu celălalt, asigurând ca ansamblu, operarea platformei țintă (Motorola DSP56307 Evaluation Module) conform principiilor TR vizate:

- BOOT – secvența care realizează încărcarea și lansarea sistemului;
- SYSINIT – task-ul de inițializare a structurilor de date și a operării nucleului;

- DATALINK – set de task-uri care implementează comunicația de date și comenzi dintre HARETICK și mediul de dezvoltare INVERTA care rulează pe un calculator gazdă;
- MONITOR – task-ul care implementează interfața HARETICK cu operatorul;
- LOADER – task-ul care realizează încărcarea unei aplicații noi în sistem și, în același timp, gestionează memoria aplicație a HARETICK;
- SSCD – planificatorul contextului lejer, preemptiv de execuție a nucleului;
- HSCD – planificatorul contextului strict, non-preemptiv de execuție;
- HDIS – executivul TR al nucleului HARETICK, cu rol în comutarea celor două contexte de execuție și încadrarea execuției ModX-urilor din sistem;
- STATREPO – task-ul care realizează raportarea stării curente a sistemului.

Timpul, ca parametru esențial de operare al sistemului, este reprezentat și gestionat cu ajutorul unor structuri și mecanisme dedicate. *Baza de timp* a sistemului este implementată de *ceasul timp-real* (RTC) – elementul activ de gestionare a timpului în HARETICK.

RTC are două roluri esențiale: (i) contor de timp, utilizat pentru referiri ale componentelor sistemului la instanța curentă de timp, și (ii) singura sursă (predictibilă) de întreruperi. RTC este o sursă predictibilă de întreruperi în sensul că se poate determina oricând, și de către orice task, momentul apariției următoarei întreruperi.

Pe lângă RTC, nucleul HARETICK gestionează încă două structuri de timp. *Timpul absolut sistem*, este utilizat pentru referiri ale operatorului și ale task-urilor din sistem la timpul absolut, și poate fi setat (modificat) de către operatorul sistemului în orice moment, prin intermediul unor comenzi specifice. *Timpul de planificare* este utilizat și gestionat de către planificatorul contextului strict de execuție al sistemului pentru calcularea și specificarea momentelor la care sunt planificate execuțiile task-urilor TR stricte.

Spre deosebire de timpul absolut sistem, celelalte două structuri – ceasul timp-real și timpul de planificare, lucrează practic în domeniu temporal relativ. Pe de altă parte, toate cele trei tipuri de timp cu care lucrează nucleul HARETICK sunt sincronizate între ele.

Task-urile TR stricte sunt modelate, în concepția lucrării de față, prin ModX-uri – task-uri periodice, modulare, cu specificații complete și stricte de comportare în timp și cu planificare și execuție în context non-preemptiv. Un ModX conține seturi de parametri de intrare și de ieșire, variabile globale, un set de parametri ce îi specifică comportarea în timp, seturile semnalelor de intrare și de ieșire cu care interacționează, și secvența de cod program (specificarea funcțională).

În cadrul nucleului HARETICK, ModX-urile sunt reprezentate cu ajutorul unor structuri și mecanisme specifice:

- "PDAGT" (Program Directed Acyclic Graph Table), tabela ce conține graful direcționat, aciclic, de program al aplicației;
- Zona de cod ("CODE") a ModX-ului;
- Zona parametrilor de ieșire ("Data Out");
- Zona variabilelor globale ale aplicației;
- Zona variabilelor locale și ai parametrilor de intrare ai ModX-ului în execuție ("Heap");

- Tabela de simboluri;
- Tabela descriptorilor de procese ("PDT").

ModX-urile sunt executate în cadrul contextului strict al HARETICK, fiind planificate de către ModX-ul sistem HSCD. Activarea și terminarea ModX-urilor este gestionată de către executivul TR al nucleului, HDIS.

La începutul execuției fiecărui ModX, are loc în mod obligatoriu operația de copiere a valorilor parametrilor de ieșire, din zonele "Data Out" ale ModX-urilor de care depinde cel curent, în locațiile parametrilor de intrare, prevăzute în "Heap". Timpii de execuție a instrucțiunilor implicate de această operație se vor adăuga la WCET-ul ModX-ului.

Comunicația inter-task-uri este realizată prin transmiterea parametrilor de intrare/ieșire, cu mecanismul descris anterior. Datorită structurii și reprezentării ModX-urilor (care, practic, implementează operații atomice), și a mecanismului de comunicație dintre ele, controlul și sincronizarea acceselor la resurse concurente nu mai este o problemă. Un alt aspect interesant legat de acest subiect îl constituie facilitatea de transmitere a stării unui ModX, de la o execuție a sa la alta, prin intermediul unui același parametru care este și de intrare, și de ieșire.

La sfârșitul execuției fiecărui ModX, se găsește instrucțiunea de apel la HDIS, care, practic înlocuiește în mod echivalent instrucțiunile utilizate uzual în acest caz (`return`, `exit`, `end`, `RTS`, etc.).

Funcționarea de principiu a sistemului se bazează pe cele două contexte de execuție complementare – contextul strict și cel lejer, și pe comutarea acestora, realizată de către executivul TR al HARETICK, HDIS.

Contextul strict de execuție poate fi privit ca o "întrerupere prelungită" a celui lejer, conform unei planificări riguroase, stricte, analizată apriori din punctul de vedere al fezabilității (analiză "offline").

Un alt mod de a vedea problema este acela că, deasupra contextului lejer (reprezentând modul de operare tradițional, preemptiv, al sistemelor de calcul) se poate implementa, ca o facilitate de execuție cu garantarea respectării termenelor stricte și cu predictibilitate maximă, contextul strict, a cărui întrerupere (RTC) este cea mai prioritară în sistem. În plus, pe durata execuției în contextul strict, celelalte întreruperi sunt suspendate.

Au fost definite 5 stări de operare a ModX-urilor în cadrul sistemului HARETICK:

- NOP ("No Operation") – descrie ModX-urile unei aplicații ce încă nu sunt încărcate în sistem.
- RDY ("Ready") – reprezintă starea în care se află ModX-urile încărcate în sistem și care așteaptă să fie planificate. În această stare, ModX-urile sunt complet identificate și reprezentate, având alocate majoritatea resurselor sistem necesare.
- SCD ("Scheduled") – descrie ModX-urile care sunt planificate în cadrul ciclului curent de planificare calculat de către HSCD. În această stare, ModX-urile se regăsesc și în înregistrările Tabelii de Dispatch.
- RUN ("În execuție") – reprezintă starea unui singur ModX, căruia i s-a alocat procesorul sistem în vederea executării, în regim non-preemptiv, a secvenței

de instrucțiuni proprii. ModX-ul în starea RUN a fost lansat de către executivul HDIS, prin identificarea lui în Tabela de Dispatch.

- GST ("Ghost") – starea ce descrie un ModX fantomă la un moment dat (ModX-ul care, în momentul lansării, are contorul de execuții nul).

Din punctul de vedere al cerințelor impuse de arhitectura nucleului HARETICK, calcularea timpului maxim de execuție a ModX-urilor (*WCET*) implică următoarele elemente:

- durata de execuție a celei mai lungi căi de program din codul propriu-zis al ModX-ului;
- durata secvenței de instrucțiuni, de la începutul execuției ModX-ului, care efectuează operațiile de copiere a parametrilor de ieșire;
- durata instrucțiunii de apel la `HDIS_SUF`, de la sfârșitul secvenței de cod a ModX-ului;
- durata maximă de execuție a executivului HDIS, cu cele două componente ale sale (prefix și sufix) și cu subrutina de raportare a evenimentelor TiLT.

8.2 Stadiul actual al implementării și perspective

Nucleul HARETICK este încă în curs de dezvoltare și testare pe platforma Motorola DSP56307. Au fost implementate majoritatea structurilor de date sistem (PDT, Tabela de Dispatch, zona de salvare/restaurare a contextului de execuție a task-urilor lejere întrerupte, tabela TiLT, zona "Heap", etc.). De asemenea, au fost definite și alocate corespunzător zonele pentru cod și parametri de ieșire ("Data Out") ale ModX-urilor.

Executivul timp-real al nucleului, HDIS, a fost complet proiectat și implementat. Testele preliminare de execuție pe porțiuni de cod pentru HDIS au fost efectuate, cu rezultate pozitive.

În prezent se implementează componentele BOOT, SYSINIT și LOADER ale nucleului, și în același timp, se lucrează la o aplicație simplă pentru a testa funcționarea întregului sistem. Aplicația va implementa un generator de semnale strict periodice, astfel încât corectitudinea de procesare și respectarea specificațiilor de operare în timp ale aplicației să poată fi verificate și din exteriorul sistemului, cu ajutorul unui osciloscop.

Preocupările de viitor imediat legate de implementarea sistemului HARETICK cuprind următoarele elemente:

- conceperea și implementarea unei structuri statice, convenabile, destinată reprezentării eficiente a grafului aplicațiilor ce vor fi încărcate în sistem (tabela PDAGT – Program Directed Acyclic Graph Table);
- finalizarea implementării tuturor componentelor nucleului HARETICK;
- înscrierea codului sistemului în memoria EPROM a platformei țintă și configurarea sistemului pentru a încărca secvența din EPROM la pornire sau după o operație de RESET;

- finalizarea implementării protocolului și interfețelor de comunicație cu calculatorul gazdă (DATALINK), pentru a se putea citi din exterior rapoarte periodice ale execuției sistemului HARETICK;
- construirea și implementarea unor aplicații, pentru testarea întregului concept;
- efectuarea diverselor analize necesare evaluării performanțelor sistemului HARETICK și determinarea modului în care arhitectura acestuia influențează proiectarea și operarea aplicațiilor TR.

În momentul rezolvării cu succes a tuturor problemelor majore legate de implementarea curentă a HARETICK, în așa fel încât sistemul să fie funcțional pe platforma Motorola DSP56307, se intenționează dezvoltarea unei noi implementări, utilizând alte tipuri de sisteme țintă (de exemplu, sisteme cu microcontroller, etc.).

Bibliografie

- [Burns 90] A. Burns, A. Wellings, "Real-Time Systems and Their Programming Languages", Addison-Wesley, 1990.
- [Coffman 73] E. G. Coffman Jr., P. J. Denning, "Operating Systems Theory", Prentice-Hall Series in Automatic Computation, Prentice-Hall, New Jersey, USA, 1973.
- [Cretu 99] V. Cretu, M. V. Micea, "Data Acquisition System - from Design to Real-Life Applications Integration", in Proceedings of the International Conference on Engineering and Modern Electric Systems EMES'99, Computer Section, University of Oradea, May 26-29, 1999, pp. (154-162).
- [Ghosh 94] K. Ghosh, B. Mukherjee, K. Schwan, "A Survey of Real-Time Operating Systems", Technical Report GIT-CC-93/18, College of Computing, Georgia Institute of Technology, Atlanta, Georgia, February 1994.
- [Jeffay 91] K. Jeffay, D. Stanat, C. Martel, "On Non-Preemptive Scheduling of Periodic and Sporadic Tasks", in Proceedings of the Twelfth IEEE Real-Time Systems Symposium, San Antonio, Texas, December 1991, IEEE Computer Society Press, pp. (129-139).
- [Kim 80] K. H. Kim, M. Naghibzadeh, "Prevention of Task Overruns in Real-Time Non-Preemptive Multiprogramming Systems", ACM, 1980, pp. (267-276).
- [Letia 00] T. S. Letia, "Real-Time Systems", ("Sisteme de timp-real"), "Editura Albastra" Publishing House, Cluj-Napoca, Romania, 2000.
- [Liu 73] C. L. Liu, J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", in Journal of the ACM, Vol. 20, No 1., January 1973, pp. (46-61).
- [Micea 00a] M. V. Micea, "Signal Acquisition and Conditioning Systems: Laboratory Workshops", ("Sisteme de achizitie numerica a datelor: Indrumator de laborator"), Comanda 270/2000, Centrul de multiplicare al Universitatii POLITEHNICA Timisoara, 2000.
- [Micea 00b] M. V. Micea, "Interfacing DAQ Systems to Digital Signal Processors", in Transactions on Automatic Control and Computer Science, Special Issue Dedicated to the Fourth International Conference on Technical Informatics,

- CONTI'2000, October 12-13, Vol. 45 (59), No. 4, "Politehnica" University of Timisoara, 2000, pp. (23-28).
- [Micea 00c] M. V. Micea, V. Cretu, D. Chiciudean, "Communication Protocols Implementation on DSP-based Systems", in Proceedings of the International Symposium on Systems Theory, SINTES 10, 10-th Edition, Automation, Computers, Electronics, May 25-26, University of Craiova, 2000, pp. (C 205-208).
- [Micea 00d] M. V. Micea, D. Chiciudean, L. Muntean, "ECP Standard Parallel Interface for DSP56300 Devices", Application Note AN2085/D Rev. 0, 11/2000, Motorola, Inc., USA, 2000.
- [Micea 00e] M. V. Micea, V. Cretu, D. Chiciudean, "Interfacing a Data Acquisition System to the DSP56303", Application Note AN2087/D Rev. 0, 12/2000, Motorola, Inc., USA, 2000.
- [Motorola 00] Motorola, Inc., "DSP56300: 24-Bit Digital Signal Processor: Family Manual", Revision 3, DSP56300FM/AD, Semiconductor Products Sector, DSP Division, Austin, USA, November 2000.
- [Motorola 95] Motorola, Inc., "DSP56000: 24-Bit Digital Signal Processor: Family Manual", DSP56KFAMUM/AD, Semiconductor Products Sector, DSP Division, Austin, USA, 1995.
- [Motorola 96] Motorola, Inc., "Motorola DSP: Assembler Reference Manual", Version 6.0, Semiconductor Products Sector, DSP Division, Austin, USA, 1996.
- [Motorola 98] Motorola, Inc., "DSP56307: 24-Bit Digital Signal Processor: User's Manual", DSP56307UM/D, Revision 0, 08/10/98, Semiconductor Products Sector, DSP Division, Austin, USA, August 1998.
- [Motorola 99] Motorola, Inc., "DSP56307EVM User's Manual", Revision 1.8, 3/1999, DSP56307EVMUM/D, Semiconductor Products Sector, DSP Division, Austin, USA, March 1999.
- [Panzieri 93] F. Panzieri, R. Davoli, "Real Time Systems: A Tutorial", Technical Report UBLCS-93-22, Laboratory of Computer Science, University of Bologna, Italy, October 1993.
- [QNX 99a] QNX Software Systems, Ltd., "QNX Neutrino v2 Realtime Operating System: Building Embedded Systems", 3rd Edition, Ontario, Canada, May 1999.
- [QNX 99b] QNX Software Systems, Ltd., "QNX Neutrino v2 Realtime Operating System: Development Tools", 2nd Edition, Ontario, Canada, June 1999.
- [Robu 02] N. Robu, "Concurrent Programming. Real-Time Oriented Support Mechanisms", ("Programare concurenta. Mecanisme

- suport orientate timp real"), Editura "Politehnica" Timisoara, Romania 2002.
- [Stallings 98] W. Stallings, "Operating Principles: Internals and Design Principles", 3rd Edition, Prentice-Hall, New Jersey, USA, 1998.
- [Stankovic 88] J. A. Stankovic, "Misconceptions About Real-time Computing", in IEEE Computer, Vol. 21, No. 10, October 1988, pp. (10-19).
- [Stewart 01] D. B. Stewart, "Twenty-five Most Common Mistakes with Real-time Software Development", in 2001 Embedded Systems Conference, Class 270, San Francisco, April 2001.
- [Tanenbaum 97] A. S. Tanenbaum, A. S. Woodhull, "Operating Systems: Design and Implementation", 2nd Edition, Prentice-Hall, New Jersey, USA, 1997.