A TERMES real-time operációsrendszer kapcsolástechnikai alkalmazásra

DR. TOLDI GÁBOR—DR. VERESS TIBOR—
BALLA GÁBOR—LAKATOS PÉTER
BHG Híradástechnikai Vállalat, Fejlesztési Intézet

ÖSSZEFOGLALÁS

A cikk a BHG Fejlesztési Intézetben kifejlesztett TERMES realtime operációs rendszert ismerteti, amely elsősorban kapcsolástechnikai alkalmazásra lett kialakítva, de egyéb célú rendszerekben is előnyösen alkalmazható. A cikkben a szerzők ismertetik a fejlesztés során figyelembe vett követelményeket, célokat, majd részletesen bemutatják az egyes funkciók megvalósítását, azok működését.

DB. TOLDI GÁBOR A Budapesti Műszaki Egyetem Villamosmérnöki Kara híradástechnika szakán 1977-ben szerzett villamosmérnöki, majd 1979-ben híradástechnikai szakmérnöki oklevelet. 1980-ban védte meg a műszaki doktori disszertációját, melynek témája a mikroprogramozott célszámítógépek tesztelési, élesztési kérdései. 1977-től a BHG Fejlesztési Intézetében dolgozik, szakterülete az elektronikus tároltprogram-vezérlésű telefonközpontok vezérlőrendszerének, azok rendszer-szoftverének fejlesztése.

Bevezetés

A BHG-ban az 1970-es évek elejétől folyik tároltprogram-vezérlésű telefonközpontok fejlesztése. A nemzetközi piacon fokozódó konkurenciaharc az egyéni, speciális vevői igények mind teljesebb és gyorsabb kielégítésére készteti a vállalatot. Ezekre a kihívásokra csak akkor tud megfelelően gyors fejlesztésekkel reagálni, ha mind következetesebben érvényesíti az egyes termékcsaládjai kialakításánál az építőkockaszemléletet, a modularitás elvét.

Másrészt a fejlesztői-alkotói elme véges áttekintőképessége következtében a magas igényeket kielégítő, egyre komplexebb berendezések fejlesztése csak úgy lehetséges, ha a tervezés egyes fázisaiban bizonyos funkciókra, mint jól definiált, meglevő építőelemekre gondolhatunk, ezáltal a problémát vertikumában tagolhatjuk. Ez a követelmény szintén a berendezések moduláris felépítése révén elégíthető ki.

Maga a modularitás elve a BHG-ban sem új, az építőkocka-szemlélet előnyeit tapasztaltuk a QA—EP alközpontcsalád kapcsán. Míg azonban a korábbi megoldások elsősorban a moduláris HW-re koncentráltak, a berendezések SW-ének komplexitása oly mértékben nőtt, hogy az építőelem-szemlélet szigorú alkalmazása ma már itt is elengedhetetlen.

Az elmúlt néhány év fejlesztésének eredménye HW-területen az új generációs mikroprocesszoros vezérlőrendszer, az EXCEL (*Ex*change *C*ontrol *El*ements). Moduláris felépítése, korszerű tervezése révén különböző méretű telefonközpontok és egyéb célú berendezések vezérlőrendszereinek kialakítására alkalmas.

Az EXCEL vezérlőrendszer különféle célú alkalmazásait elősegítendő, fejlesztettük ki a TERMES (*T*elephon *E*xchange *R*eal-time *M*ultitasking *E*xecutive *S*ystem) operációs rendszert, amely moduláris felépítése, jól definiált interfészei, építőkockajellege révén nagymértékben támogatja a moduláris SW-rendszerek kialakítását.

A cikk további részében a TERMES operácósi rendszerrel foglalkozunk.

Beérkezett: 1986. IV. 2. (#)

Fejlesztési célkitűzések

A TERMES elsősorban azzal a céllal készült, hogy a legkülönfélébb telefonközpontok EXCEL-alapú vezérlőrendszereiben az operációsrendszer-funkciókat ellássa. Tehát egy real-time folyamatirányító rendszerről van szó, amelynek a kapcsolástechnikáról fakadóan sok (100-as nagyságrend) folyamatot kell tudnia párhuzamosan kezelnie, szigorú real-time-követelmények mellett.

A fejlesztésnél természetesen szem előtt tartottuk az egyéb (nem kapcsolástechnikai) célú rendszerekben történő alkalmazhatóságot. A rendszernek kellően flexibilisnek, az adott igényeknek megfelelően konfigurálhatónak kellett lennie.

A rendszernek nyíltnak kell lennie abban az értelemben, hogy további funkciók, mint az operációs rendszer külső héjai, könnyen ráépíthetők legyenek.

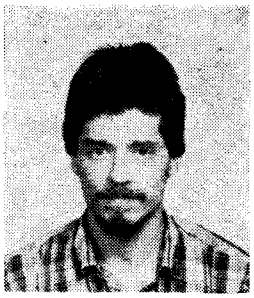
A rendszernek illeszkednie kell a jelenlegi EXCEL-vezérlők HW-specialitásaihoz: pl. 8085 mikroprocesszor-típus.

A rendszernek támogatnia kell az alkalmazói programok fejlesztését. Kívánatos volna a meglevő mikroprocesszoros fejlesztői környezet alkalmazhatósága, ezen belül a PL/M—80 magasszintű programnyelvvel való kompatibilitás biztosítása.

Végül, de nem utolsósorban, a bevezetés szellemének megfelelően, a rendszernek szigorúan moduláris felépítésűnek kellett lennie. Az egyes moduloknak jól definiált és lehetőleg egyszerű belső és külső interfészekkel kell rendelkezniük. A modulokra bontás akkor jó, ha az egyes modulok jól definiált, egyszerűen leírható funkciókat valósítanak meg. Az egyes modulok az adott funkciót lehetőleg teljesen fedjék le, mert különben elkerülhetetlen az azonos, vagy hasonló célú modulok és nem moduláris elemek megjelenése a rendszerekben. Másrészt a modulok lehetőleg egyszerű funkciókat valósítanak meg, mert ellenkező esetben a modulok túl komplexszé válnak, ami az alkalmazások többségében felesleges terhet jelent. Ezért a modulokra bontásra, az egyes modulok funkcióinak lehatárolására különös figyelmet kell fordítani.

DB. VERESS TIBOR

A Budapesti Műszaki Egyetem Villamosmérnöki Kara híradástechnika szakán 1979-ben villamosmérnöki, majd 1981-ben számítástechnikai szakmérnöki oklevelet szerzett. „Hibatűrő irányító rendszerek a kapcsolástechnikában" című doktori disszertációját 1982-ben védte meg. 1979 óta a BHO Fejlesztési Intézetben, a Vezérlőfejlesztési Osztályon dolgozik. Szakterülete: elektronikus tároltprogram-vezérlésű telefonközpontok vezérlésének, fejlesztői környezetének fejlesztése.



A TERMES főbb funkciói

A TERMES olyan folyamatorientált operációs rendszer, ahol az egyes folyamatokat realizáló taskok számára egy-egy virtuális processzort (6) biztosít a rendszer. Így az egyes folyamatok nagymértékben függetlenek lehetnek egymástól, a taskok úgy „észlelik", mintha a processzor kizárólag a sajátjuk lenne. Így a taskok programjai nem bonyolódnak azáltal, hogy multitasking környezetbe kerülnek.

A TERMES főbb funkciói a következők:

a) A legfontosabb: a kezelendő folyamatok számára létrehozza a virtuális processzorokat. Az ún. ütemezési algoritmus alapján a fizikai CPU-időt szétosztja ezek, azaz a taskok között, ezzel biztosítja a folyamatok kváziparalel futását (Scheduler).

b) A TERMES megoldást nyújt a folyamatok futásának összehangolására, szinkronizációjára (Semaphore Manager).

c) A TERMES eszközt biztosít a folyamatok kommunikációs kapcsolatának kialakítására is (Mailbox Manager).

d) A TERMES egységes módszert biztosít a külső környezet és a folyamatok kapcsolatának realizálására (Interrupt Manager).

A négy alapfunkció mellett feltüntettük a megfelelő TERMES-modul nevét is. Az alapszolgáltatások köre megfelel a real-time-rendszerek szokásos funkcióinak (1, 2).

A TERMES moduljai

A TERMES egy object-orientált rendszer. Ez azt jelenti, hogy az egyes funkciói adott struktúrájú object-típusokon értelmezett műveletek formájában valósulnak meg, melyek egyedüli eszközei az objecteken történő operációknak.

A TERMES-ben a funkciók megvalósításához ötféle object-típus létezik: a task, a semaphore, a mailbox, a segment és az interrupt exchange objectek. Ezek struktúráját az egyes moduloknál ismertetjük.

Minden object-típushoz rögzítve van a rajta értelmezett műveletek köre. Ezeket a műveleteket operációsrendszer-rutinok valósítják meg. Ezek a rendszerrutinok formailag PL/M-kompatibilis eljárások (procedure), melyek az applikációs programok számára hozzáférhetők, hívhatók (rendszerhívások).

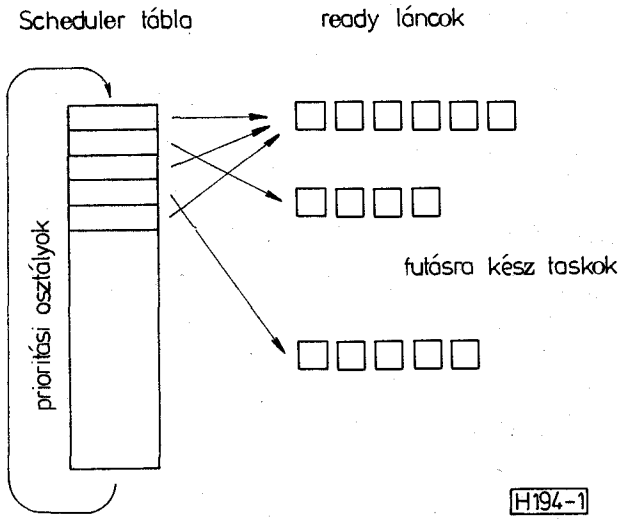
A következőkben a TERMES egyes moduljai kerülnek ismertetésre.

A Scheduler

A TERMES-ben a folyamatok párhuzamos folyása látszólagos, hiszen egy processzoron fut a rendszer, szigorúan véve egyszerre csak egy folyamat fut. A látszólagos párhuzamosság abból adódik, hogy a fizikai CPU csak rövid ideig rendelődik az egyes folyamatokhoz, azokat felváltva hajtja végre. Másképpen fogalmazva: az egyes folyamatok számára a rendszer egy-egy virtuális processzort (task) biztosít az aktuális állapotok megőrzése céljából. A virtuális processzorokhoz felváltva, rövid ideig fizikai CPU-t biztosít. Ilyenkor a virtuális processzor állapota a fizikai CPU-ba töltődik, a folyamat ténylegesen futni kezd. Adott idő elteltével (vagy ha a folyamat várakozni kényszerül) a rendszer a fizikai CPU új állapotát elmenti a virtuális processzorba, a CPU másik virtuális processzorhoz (taskhoz) rendelődik stb.

A Scheduler feladata a fizikai CPU-idő elosztása a versengő taskok között. Ez az ún. ütemezési algoritmus alapján történik. A rendszerben maximum 255 task lehet. A taskok mindegyike egy prioritási attributummal van ellátva, ami hozzávetőlegesen a taskok CPU-időért folytatott versengésének esélyeit fejezi ki. A rendszerben maximum 255-féle prioritási osztály létezhet, ez a felhasználó kezében lévő konfigurálási paraméter. Azt, hogy az egyes prioritási osztályokba tartozás ténylegesen mekkora előnyt vagy hátrányt jelent, a Scheduler-tábla fejezi ki. A Scheduler-tábla hossza és kitöltése szabad konfigurálási paraméter.

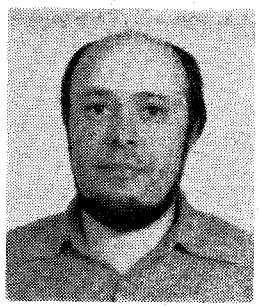
A Scheduler adott időközönként (rendszeróra, konfigurálási paraméter), valamint akkor, ha az éppen futó task várakozni kényszerül, ütemezést végez. Ilyenkor a Scheduler-tábla soron következő sorából állapítja meg azt a prioritási osztályt, amelyből a soron következő, futásra kész taskot elindítja. A futásra kész taskok — prioritási osztályok szerint csoportosítva — az ún. ready-láncokra felfűzve várakoznak a CPU-időre, azaz, hogy a Scheduler ütemezze őket *(l. 1. ábra)*. Ha egy üte-



1. ábra. Ütemezési séma

BALLA GÁBOR

A BHG Híradástechnikai Vállalat fejlesztőmérnöke. Tanulmányait a Budapesti Műszaki Egyetem Villamosmérnöki Karán, híradástechnika szakon végezte, ahol 1976-ban adat- és távközléstechnika ágazaton szerzett oklevelet. Ettől az évtől kezdve a BHG-ban dolgozik. Tároltprogram-vezérlésű telefonalközpontok vezérlőegységének HW-fejlesztési munkáiban és gyártásbavitelében vett részt. Az utóbbi időben a vezérléssel kapcsolatos rendszerszoftverrel, valamint a tároltprogram-vezérlésű telefonközpontok HW—SW fejlesztői környezetével is foglalkozik.



mezett prioritási osztály readylánca üres, azaz az adott osztályban nincs futásra kész task, a Scheduler az eggyel alacsonyabb prioritási osztályból választ. A ready-láncok lényegében FI—FO (First In, First Out) jellegű várakozási sorok.

A felhasználó a Scheduler-tábla megfelelő kitöltésével befolyásolhatja az egyes prioritási osztályok ütemezésének relatív gyakoriságát. Ez nagyfokú rugalmasságot jelent. Példaképpen három egyszerű esetet mutatunk be:

a) Egyszerű prioritásos rendszert kapunk, ha a Scheduler-tábla egyetlen sorába a legmagasabb prioritási osztályt írjuk és minden tasknak más prioritási attribútumot adunk. Így az alacsonyabb prioritású taskok csak akkor futhatnak, ha a magasabb prioritású taskok között nincs futásra kész.

b) Az egyszerű „forgó" (Round robin) rendszerhez jutunk, ha az összes tasknak azonos prioritást adunk. Ilyenkor a legrégebben a CPU-ra várakozó taskot ütemezi a Scheduler.

c) Nagyjából a bináris sornak megfelelő ütemezést nyerünk, ha a Scheduler-tábla minden második sorába a legmagasabb, minden közbülső negyedik sorába az eggyel alacsonyabb, stb. prioritási osztályt írjuk. Így az eggyel alacsonyabb prioritási osztály kb. feleakkora eséllyel jut CPU-időhöz, mint a magasabb.

A Scheduler másik fontos feladata a taskok időzített várakozásának kezelése. Az időzített taskok az időzítés lejártának abszolút időpontja szerint rendezett ún. time-láncon felfűzve várakoznak. A taskok ilyenkor nem szerepelnek a ready-láncon, így nem ütemeződnek.

A Scheduler minden ütemezéskor (pontosabban óraütéskor) összehasonlítja a 16-bites rendszeróra állását az első várakozó task abszolút várakozási időpontjával. Ha az adott időpont elmúlt, a task lekerül a time-láncról és a prioritásának megfelelő ready-láncra fűződik, így a továbbiakban részt vesz a CPU-időért folytatott versenyben.

A rendszerben egynél több time-lánc is létezhet (konfigurálási paraméter). Ily módon a time-láncok hossza mérsékelhető, a rendezett láncra való felfűzéssel járó időveszteségek csökkenthetők.

A rendszerben az óraütés tulajdonképpen egy periodikus interrupt, amely a Schedulert működteti. Ennek a periódusa konfigurálási paraméter; minél kisebb, annál jobb a rendszer időfelbontása, de annál nagyobbak a Scheduler futtatásával járó relatív többletterhek (overhead). Célszerű minimumnak látszik a kb. 10 msec-es periódus, ami a telefontechnikai alkalmazásokban is előnyös.

A Task Manager

A Task Manager a task objecttel és a Schedulerrel kapcsolatos rendszerrutinokat tartalmazza.

A megvalósított funkciók:

— a task aktuális prioritásának lekérdezése (TS$GPRI)
— a taskok aktuális prioritásának megváltoztatása (TS$SPRI)
— a task azonosító címének (token) lekérdezése (TS$TOKN)
— a task várakoztatása egy adott abszolút időpontig, azaz a 16-bites rendszeróra egy adott állásáig (TS$WAIT)
— a task késleltetése egy relatív időtartamig, azaz valahány óraütés elteltéig (TS$DLAY)
— egy tetszőleges task futásának felfüggesztése (TS$SUSP)
— a felfüggesztett task újraélesztése (TS$RESM)

A task object a task exchange-láncon (l. később) való elhelyezésére szolgáló láncparamétereket, aktuális és maximális prioritási attributumot, a task állapotát (futásra kész, időzített, várakozik, felfüggesztett stb.) tartalmazza.

A Semaphore Manager

A Semaphore Manager funkciója a taskok szinkronizációjának megvalósítására. A folyamatok közti legalapvetőbb szinkronizációs alapsémák a kölcsönös kizárás (pl. közös erőforrásokhoz való kizárólagos hozzáférés), a fogyasztó—termelő séma (pl. pufferkezelés) és az eseményjelzés (pl. adott jelre egyszerre történő indítás) típusú szinkronizáció. Ezek tárgyalása a szakirodalomban bőséges (4, 5).

A TERMES Semaphore Managere mindhárom alapséma funkcióit nyújtja. Az egyes funkciók egy ún. semaphore objecten értelmezett műveletként valósulnak meg. A semaphore object lényegében egy 16-bites számlálóból és egy exchange-lánc listafejből áll. A számláló egy elvonatkoztatott erőforrás rendelkezésre álló darabszámát tartja nyilván. A taskok igényelhetnek a rendelkezésre álló erőforrásokból, illetve a már igényelt erőforrásokat visszajuttathatják, felszabadíthatják. Ha egy task annyi absztrakt erőforrást igényel, amennyi a semaphore szerint pillanatnyilag nem áll rendelkezésre, a task a semaphore exchange-láncára felfűződik és várakozó állapotba kerül, míg egy másik task fel nem szabadítja a kívánt mennyiségű erőforrást az adott semaphore-on.

Az exchange-láncok szervezése FI—FO jellegű, azaz a taskok „érkezésük" sorrendjében szolgálódnak ki. A semaphore-on történő várakozás időben korlátozható.

A megvalósított műveletek:

— absztrakt erőforrások felszabadítása (SF$SEND)
— absztrakt erőforrások igénylése időkorlát nélküli várakozással (SF$RCIW)

— absztrakt erőforrások igénylése abszolút időkorláttal (SF$RCAW)
— absztrakt erőforrások igénylése relatív időkorláttal (SF$RCRW)
— absztrakt erőforrások igénylése várakozás nélkül (SF$RCNW)
— a semaphore számlálójának iniciális értékkel való feltöltése, egyben az összes várakozó task felszabadítása (SF$INIT)

A fentiekben ismertetett semaphore objeettel és műveleteivel mindhárom szinkronizációs alapséma megvalósítható: a fogyasztó-termelő modell magától értetődő; a kölcsönös kizárás alapesetét kapjuk, ha I absztrakt erőforrással dolgozunk; az esemény jelzésére pedig az SF$INIT műveletrévén van mód.

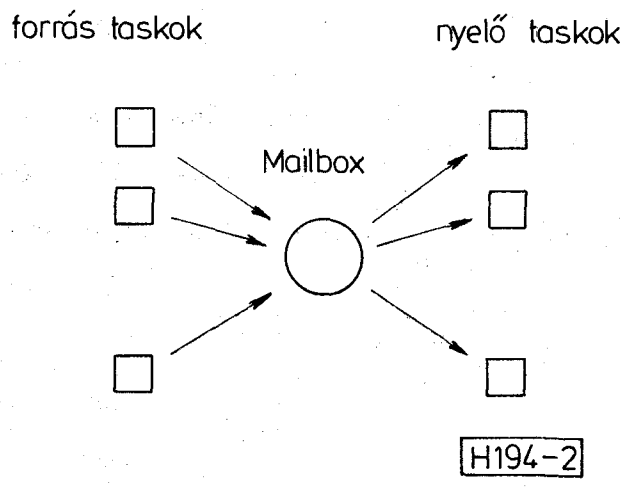
A rendszerben lévő semaphore-oknak a számát csak a rendelkezésre álló memória korlátozza.

A Mailbox Manager

A Mailbox Manager modul feladata a taskok processzoron belüli kommunikációjának lebonyolítása. Ez a funkció a TERMES segment és mailbox objectjei és ezek műveletei révén valósulnak meg.

A segment object lényegében egy láncparaméterekkel ellátott, tetszőleges hosszúságú memóriadarab, amelyben tetszőleges információ helyezhető el. Az információforrás szerepű task, miután az átadandó információval kitöltött egy segment objectet, a megfelelő művelettel egy mailbox objecten helyezi el azt. Az információnyelő szerepű task pedig az adott mailbox objecten jelentkezik a segment objectért és átveszi azt, azután kiértékelheti az „üzenetet". A forrás és nyelő taskok futásának aszinkronitását oldja fel, hogy a mailbox objecten akárhány segment object üzenet elhelyezhető anélkül, hogy az előzőeket átvenné egy nyelő task. Ilyenkor a segment objectek egy ún. object-láncra felfűzve várakoznak a nyelő task jelentkezésére. Ugyanígy, ha a nyelő task(ok) túl korán jelentkeznek és még nincs a mailboxon elhelyezett segment üzenet, a task(ok) várakozni kényszerülnek az adott mailbox exchange-láncára felfűződve. A segment objectek elhelyezésével a várakozó nyelő taskok felszabadulnak.

A mailboxon való várakozás időben korlátozható, ahogy a semaphore-nál láttuk. Egy mailboxra akárhány forrás és nyelő task dolgozhat *(l. 2. ábra)*.



2. ábra. Taskok kommunikációja mailboxon keresztül

LAKATOS PÉTER
A BHG Fejlesztési Intézet fejlesztőmérnöke. 1981-ben a Budapesti Műszaki Egyetem Villamosmérnöki Karán, híradástechnika szakon szerzett oklevelet. Ettől az évtől kezdve a BHG-ban dolgozik. Kezdetben a tároltprogram-vezérlésű telefonközpontok vezérlőrendszereinek fejlesztésében vett részt, később tevékenysége a kommunikációs hálózatok területére koncentrálódott. Az utóbbi időben elsősorban a vezérlőrendszerek operációs rendszerének és kommunikációs szotverének fejlesztésével foglalkozik.

A rendszer nem biztosítja nyelő oldalon a források (feladók) azonosítását, sem a küldemények szortírozását. A rendszerben azonban (a memóriakorláton belül) akárhány mailbox és segment object létezhet.

A megvalósított műveletek:
— Adott segment object elhelyezése adott mailbox objecten (MB$SEND)
— Egy segment átvétele adott mailboxról korlátlan várakozással (MB$RCIW)
— Egy segment átvétele adott mailboxról abszolút időkorláttal (MB$RCAW)
— Egy segment átvétele adott mailboxról relatív időkorláttal (MB$RCRW)
— Egy segment átvétele adott mailboxról várakozás nélkül (MB$RCNW)

A mailbox object lényegében egy exchange- és egy object-lánc listafejből áll. Az exchange-láncon a korán érkezett nyelő taskok várakozhatnak, az object-láncon pedig a még át nem vett segment objectek. A két lánc közül az egyik mindig üres.

Az Interrupt Manager

Az Interrupt Manager feladata a rendszer taskjainak és a külvilág jelzéseinek összekapcsolása, szinkronizálása. A külső jelzések (interruptok) hatására a rendszerben IT-rutinok indulnak. Ezek nem taskok, nem ütemeződnek, nem hívhatnak rendszerrutinokat (egy speciális kivételével). Futási idejük célszerűen rövid, lényegesen kisebb, mint a rendszeróra periódusa. Az IT primér feldolgozását elvégezhetik, azonban, ha hosszadalmasabb feldolgozásra van szükség, task(ok)kal kell kapcsolatba lépjenek. Ez az IT exchange object mechanizmusán keresztül lehetséges.

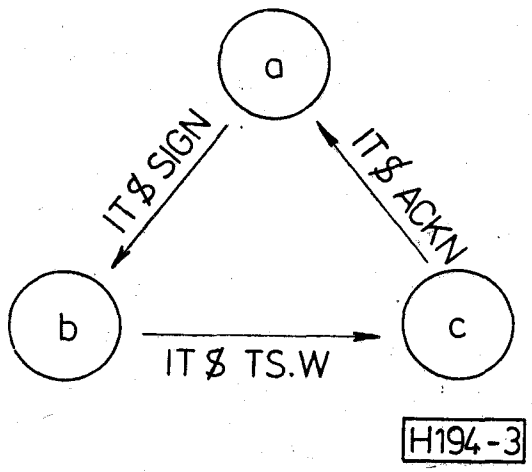
Az IT exchange egy a semaphore-hoz hasonló működésű object, amelynek a számlálója egy hipotetikus (absztrakt) I/O buffer telítettségét tartja számon. Lényegében egy fogyasztó-termelő jellegű szinkronizációt végez az IT-rutin és a feldolgozó task között. Az absztrakt működési mechanizmus mind input (IT-rutintól a task felé), mind output (task felől az IT-rutin felé) irányú információ- (feladat) áramlásra adaptálható a műveletek megfelelő értelmezésével.

A szinkronizáció úgy valósul meg, hogy ha az IT-rutin a feldolgozásban előreszalad (input esetén megtelik, output esetén kiürül a buffer) az IT automatikusan letiltódik. Ha a feldolgozó task a gyor-

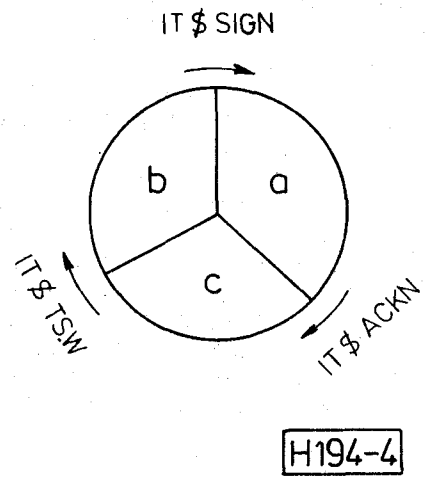
sabb (input esetben kiürül, output esetben betelik a buffer) előbb-utóbb várakozásra kényszerül az IT exchange-en. Ha a feldolgozó partner (IT-rutin, ill. a feldolgozó task) behozza a lemaradását, a letiltott IT engedélyeződik, illetve a várakozó task felszabadul. Az IT exchange számlálója, amely a hipotetikus buffer telítettségét (input esetben), vagy ürültségót (output esetben) tartja számon, végértékkel (ami a buffer hosszának felel meg) valamint alsó és felső margóértékekkel van ellátva, melyek átlépése (a buffer telítődése, ill. kiürülése esetén) okozza az IT-t keltő folyamat megfékezését, újraindítását, végső esetben az IT letiltását, illetve engedélyezését. A végérték, az alsó és felső margó értéke konfigurálási paraméter az 1—255 tartományban.

Az Interrupt Manager szempontjából a hipotetikus buffer egyes sorai háromféle állapotban lehetnek a működés során:

a) az IT-rutin hatáskörében, az IT-rutin feldolgozására várva
b) az IT-rutin hatáskörében, de már a rutin által feldolgozva
c) a task hatáskörében, feldolgozás alatt

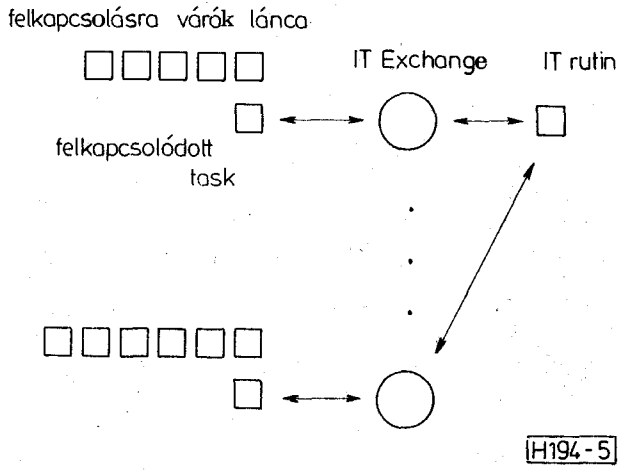


3. ábra. Az absztrakt I/O-buffer állapot-átmenetei



4. ábra. Az absztrkt I/O-buffer tartományainak változása az egyes műveletek hatására

A működés során a három állapot ciklikusan váltja egymást az IT-exchange műveletei hatására *(l. 3. ábra)*. Ennek megfelelően a buffert három tartományra oszthatjuk. A megértést segíti, ha a buffert egy körrel jelöljük *(l. 4. ábra)*, amelynek három cikkje jelenti a három külön állapotban lévő tartományt. A tartományhatárok, melyeket a konkrét alkalmazásban pointerek jelölhetnek ki a bufferben, az egyes műveletek hatására mozdulnak el egy buffersorpozíciónyit a nyíllal jelölt irányban. Az egyes tartományok nagysága a működés során — a task és az IT-rutin feldolgozási sebességétől függően — változhat.



5. ábra. IT-rutin és IT-taskkapcsolata

Egy IT-rutin több IT-exchange-en keresztül több taskkal is kapcsolatban állhat. Egy IT-exchange-hez azonban csak egy IT-rutin és adott pillanatban csak egy task tartozhat. A taskok mielőtt az IT-rutinnal kapcsolatba kerülnének, felkapcsolódnak az adott IT-exchange-hez, ezzel biztosítva a többi taskkal szembeni kizárólagos hozzáférést. A felkapcsolódott task, egy lekapcsolódási művelettel megszüntetheti az IT-rutinnal való kapcsolatát, ezzel más taskokat engedhet az IT-rutinhoz. Mivel egyszerre csak egy task lehet felkapcsolódva az adott IT-exchange-en, az IT-rutinra várakozó taskot nem kell felláncolni, nem képződhet várakozási sor. Az IT-exchange exchange-láncára a felkapcsolódásra váró taskok láncolódnak fel *(l. 5. ábra)*.

A megvalósított műveletek:

— Felkapcsolódás korlátlan várakozással (IT$CNIW)
— Felkapcsolódási kérelem abszolút időkorláttal (IT$CNAW)
— Felkapcsolódási kérelem relatív időkorláttal (IT$CNRW)
— Felkapcsolódási kérelem várakozás nélkül (IT$CNNW)
— Az IT letiltása (IT$DISA)
— Az IT engedélyezése (IT$ENAB)
— Egy bufferpozíció átvétele az IT-rutintól várakozás nélkül (IT$TSNW)
— Egy bufferpozíció átvétele az IT-rutintól korlátlan várakozással (IT$TSIW)

— Egy bufferpozíció átadása az IT-rutinnak (IT$ACKN). Az alsó margó átlépése esetén automatikusan engedélyezi az IT-t.

Csak az IT-rutinból hívható:

— Egy bufferpozíció feldolgozásának jelzése (IT$SIGN). A felső margó átlépése, illetve a végérték elérése esetén automatikusan letiltja az IT-t.

Az IT exchange object egy exchange-lánc listafejet, 8-bites számlálókat, azok végértékét, alsó és felső margóértékét, egy állapotszót, a felkapcsolódott task azonosítócímét (token), valamint a felhasználó által szolgáltatott IT-letiltó és -engedélyező rutinok belépési címét tartalmazza. A rendszerben a memóriakorláton belül akárhány IT exchange lehet.

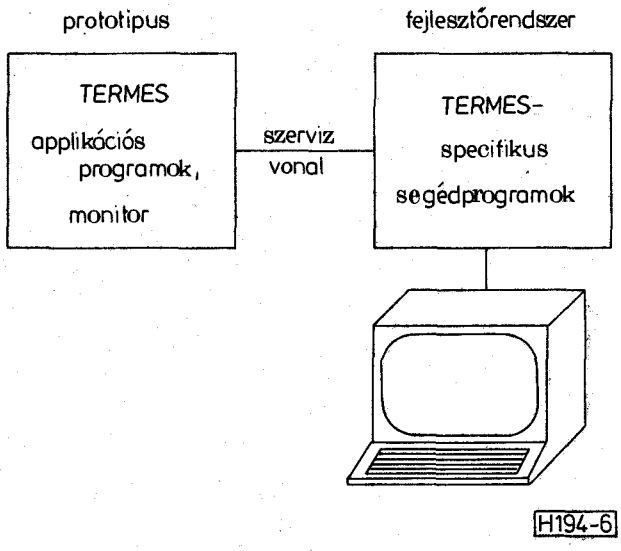
A Wboot

A TERMES egy úgynevezett statikus operációs rendszer. Ez azt jelenti, hogy szemben a dinamikus rendszerekkel, az objectek kizárólag fordítási időben definiálódnak, futási időben azok létrehozására, megszüntetésére nincs lehetőség.

A Wboot feladata, hogy a fordítási időben definiált objecteket a tényleges futás előtt RAM-területen létrehozza, kitöltse, a taskokat a megfelelő ready-láncra felfűzze. Ily módon a TERMES EPROM-bázisú rendszerekben is működőképes. A Wboot számára a felhasználó egy macronyelven köteles leírni az objectjeit, az ebből generált kódot dolgozza fel a Wboot, ez alapján tölti ki az objecteket.

A felhasználói programok

A felhasználói programok célszerűen PL/M- vagy assembly-nyelven íródhatnak. A taskok programjai vég nélküli PL/M-eljárás formáját öltik, melyeknek paramétereik is lehetnek. Maga az eljárástörzs programozása olyan, mint a hagyományos, szekvenciális programozás esetén, mintha a teljes CPU csak azt a programot hajtaná végre. Lehetőség van közös programú taskok reentrant programozására is, ez a kapcsolástechnikában különösen fontos.



6. ábra. A felhasználói programok fejlesztése

A felhasználónak célszerűen egy külön modulban definiálnia kell az objecteit. Ezt egy speciális macronyelven írhatja le. Ezután a felhasználói modulok (beleértve az objectdefiníciós modult is) összelinkelendők a szükséges TERMES-modulokkal. Az előállt objectkód EPROM-ba égethető, vagy vizsgálat céljából RAM-területre tölthető.

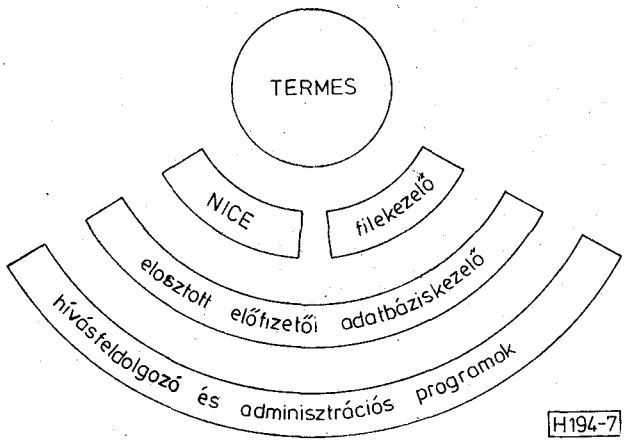
A felhasználói programok belövését egy egyszerű rezidens monitorprogram és a fejlesztőrendszer segédprogramjai támogatják. A monitorprogram a TERMES-rendszerrel konform, a kettő egymás mellett működőképes, így a futó TERMES-rendszer is vizsgálható a monitor segítségével. A monitor az EXCEL-vezérlők soros szervizvonalán keresztül kommunikál a fejlesztőrendszerrel, ahol a segédprogramok jelenítik meg a monitor üzeneteit *(l. 6. ábra)*. Ezek a segédprogramok a rezidens monitor szolgáltatásait bővítik, lehetőséget adhatnak pl. szimbolikus vagy taskspecifikus vizsgálatokra.

A felhasználói programok belövése után a monitor kihagyható a rendszerből.

Továbbfejlesztési lehetőségek

A TERMES nyitott rendszer. Szolgáltatásaira — mint nucleusra — magasabb szintű funkciókat megvalósító rétegek építhetők. Objectjeiből összetett, ún. composite objectek definiálhatók, melyeken értelmezett műveletek az elemi objectek műveleteinek felhasználásával definiálhatók. Az így kialakított rétegek mint az operációs rendszer külső héjai funkcionálhatnak.

Ilyen külső héjként lett kialakítva egy processzorközti kommunikációt megvalósító programcsomag, a NICE, amely az EXCEL-rendszer lokális hálózatán (SERBUS) teszi lehetővé különböző processzorok taskjainak kommunikációját.



7. ábra. Példa a TERMES és a különböző szintű felhasználói programok kapcsolatára

De ugyanígy külső héjként valósítható meg pl. egy file-kezelő rendszer, sőt erre és az előbbi kommunikációs héjra további héj építhető: pl. egy többprocesszoros, elosztott előfizetői adatbáziskezelő rendszer *(l. 7. ábra)*.

Értékelés

A modularitás elvének következetes betartásával, az objectszemlélet érvényesítésével real-time operációsrendszer-építőkockához jutottunk, amely real-time-rendszerek fejlesztésének meggyorsítását teszi lehetővé azáltal, hogy kész alkotóelemként beépítve kipróbált megoldásokat, eszközöket nyújt a real-time-rendszerek leggyakoribb problémáira. Moduláris felépítése, jól definiált interfészei révén elősegíti a moduláris felhasználói rendszerek kialakítását. Flexibilitása, megvalósított funkcióinak általánossága révén széleskörűen alkalmazható mind kapcsolástechnikai, mind egyéb célú real-time-rendszerekben. Egyes speciális vonásai, mint pl. a nagyszámú folyamat (task) paralel kezelése, közös programú taskok könnyű megvalósíthatósága, a nagy időbeli felbontás, 10 msec-es ütemezés különösen alkalmassá teszik telefonközpontok vezérlési céljaira.

IRODALOM

[1] IRMX/80 User's Guide, INTEL
[2] IRMX/86 Nucleus Rererence Manual, INTEL
[3] Getting Started With IRMX/86 System Manual, INTEL
[4] Strukturált programozás, Dahl, Dijkstra, Hoare, Műszaki Könyvkiadó, 1978
[5] Rendszerprogramok elmélete és gyakorlata, *Varga László*, Akadémiai Kiadó, 1978
[6] Tárolt programvezérlésű telefonközpontok operációs rendszere *dr. Kóczy T. László*, Híradástechnika, 1985. No. 9.
[7] Operating Systems, *J. Lorin, H. M. Deitel*, Addison Wesley Publishing Company, 1981
[8] Multiprocessing Operating System Allows Real-time Response From 16 Bit Computers, Data General Corp. Computer Design, 1982. VI.
[9] 68000 OS. Kernel is Implemented in Silicon Software Componenets Group, Computer Design, 1982. VI.
[10] A New Software Architecture for Switching Systems, D. *A. Lawson*, IEEE Transactions On Communications, 1982. VI. V Com—30 N. 6.
[11] A Composite Software Design For The Electronic Switching System, *K. Futami, T. Ota, T. Hara*, IEEE Transactions on Communications, 1982. VI. V Con-—30 N. 6.
[12] The Software Architecture For a Large Telephone Switch, *B. K. Penney, J. W. J. Williams*, IEEE Transactions On Communications, 1982. VI. V Com—30 N. 6.
[13] Software Structure of No. 5..ESS — A Distributed Telephone Switching System, T. *Duncan, W. H. Huen*, IEEE Transactions On Communications, 1982. VI. C Com—30 N. 6.
[14] Operating System Resides In Silicon, Software Components Group, Electronics, 1982. IV. 7. Vol 55 No. 7.
[15] 8086 Gets UNIX Like Operating System, Mark Williams Co. Electronics, 1982. IV. 7. Vol 55 No. 7.