Főszerkesztő: HORVÁTH IMRE
Szerkesztő: ANGYAL LÁSZLÓ

SZERKESZTŐBIZOTTSÁG

BHG	ORION	TERTA
Berecz Frigyes	Jakubik Béla	Bánsághi Pál
Bernhardt Richárd	Baracs Sándor	Baján Tibor
Eisler Péter	Csernoch János	Benedek Elek
Dr. Gosztony Géza	Froemel Károly	Egerszegi Béla
Honti Ottó	Sass Károly	Hutter Mihály
Klug Miklós	Szabó Károly	
Tölgyesi László		

BHG ORION TERTA
MŰSZAKI KÖZLEMÉNYEK
XXVI. évfolyam 1980 11. szám

A QA96 programvezérlő rendszere

MAKAY ATTILA
BHG

Bevezetés

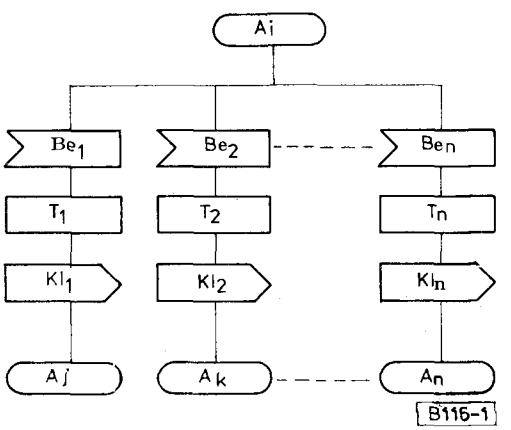
A számítógépes vagy telefonos szakmai nevén *tárolt programvezérlés* már idestova 15 éve a modern telefonközpontok egyik meghatározó jellemzője, bár kisebb kapacitású berendezésekben (néhány száz vonalig) csak az elmúlt években nyert létjogosultságot az LSI áramkörök elterjedésének eredményeképpen. A BHG-ban még a mikroprocesszorok hazai megjelenése előtt elkezdődött a kis- és közepes kapacitású kvázielektronikus telefonközpontok fejlesztése, aminek egyik részeredménye volt a speciális utasításkészlettel, egyébként egy mai átlagos 8 bites mikroszámítógéppel összehasonlítható teljesítőképességgel rendelkező vezérlőegység (MAT 512) kifejlesztése. Ez a vezérlőegység irányítja a BHG első elektronikus alközpontjának, az elmúlt év óta sorozatban gyártott QA96-nak a működését. Az alábbiakban a központ hívásfeldolgozó rendszeréről, a software felépítéséről lesz szó.

1. Általános jellemzők

A real-time környezetben lejátszódó folyamatok vezérlési feladatainak programozására kialakult módszereket a hagyományos, szekvenciális programozásban megszokottakkal szembeállítva a „real-time programozás" címszó alatt foglalják össze. Bár az elmúlt években jelentős eredmények születtek, az említett konvencionális programozás területén már jóideje elfogadott „strukturált programozás"-hoz fogható általános módszertana a real-time programozásnak még ma nincsen [3]. A ma rendelkezésre álló többé-kevésbé elfogadott módszerek valamilyen, a szóban forgó folyamatok leírására jól alkalmazható logikai modellen alapszanak [4]. Kawashima híres cikke [1] óta ma a legtöbben a véges automatákat választják a telefonközpontok működésének logikai modelljéül, és ezzel összhangban a vezérlési feladatok strukturálására a hívásosztás elvét (call division, szemben a time-division és a function-division elvével, lásd [2]-ben). Ezek alapján egy telefonközpont logikai működése általában a következőképpen írható le:

Egy telefonközpontban egyidejűleg sok egymástól független hívás megy végbe, melyek mindegyikét egy-egy véges automata jellemzi. Ezek a hívások pillanatnyi állapotait jellemző, jól definiált állapotokat vehetnek fel úgy, hogy az egyik állapotból a másikba külső jelek (legtöbbször a vonalakról kapott jelek) hatására mennek át. A jelek statisztikusan, minden belső eseményhez képest aszinkron módon lépnek fel, és a jelzések közötti időben az illető hívást „hordozó" automata stabil állapotban van (általában nem végez tevékenységet). A jelzések hatására bekövetkező állapotátmenettel együttjár a központban valamilyen meghatározott tevékenység, akció. Az automaták egy állapotát ismert módon úgy jellemezhetjük, hogy megadjuk, milyen külső jelre (BE) milyen állapotba megy át, ezen átmenet közben milyen tevékenységet (T) végez és esetleg milyen újabb, más által érzékelendő kimenő jeleket (KI) állít elő. Ezek grafikus ábrázolására, speciálisan a telefonosok igényeit figyelembe véve a CCITT dolgozott ki egyszerű módszert [5], az állapotátmeneti diagramok elvére épülő grafikus nyelvet, az SDL-t. (Specification and Description Language.) Ennek általános formája az *1. ábrán* látható.

A tényleges realizálásnál az „egy hívás — egy automata" modell fő problémája a konkrét állapotok igen nagy száma, valamint a sok parallel fo-



1. ábra. Állapotátmenet leírása SDL nyelven

lyamat működtetéséhez szükséges „operációs rendszerrel" való együttműködés leírása. Ez utóbbi feloldására a telefonközpont funkcióinak leírása céljából a hívás automaták mellé egyéb automatákat is bevezetnek [4] és így a hívások lebonyolítását több automata együttműködésével írják le (ilyen automata írhatja le magát az operációs rendszert, de más közös „funkciót" is, mint pl. a linkkeresés vagy kapcsolófokozat működtetése stb.). A CCITT—SDL egyik fő erénye, hogy le tudja írni a különböző funkcionális blokkok (folyamatok) együttműködését, jelzések adásának és vételének formájában, ahol egyébként a jelek természete nincs meghatározva.

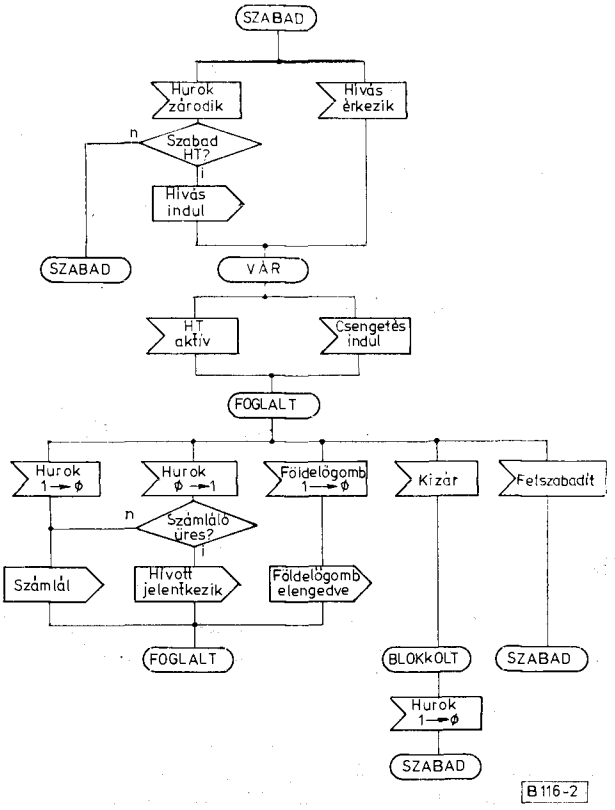
Az állapotok számának csökkentésére ugyancsak alkalmazható a fenti elv, amennyiben az automatán belül funkcionális osztást hajtunk végre és így egyazon hívást önmagában is több automata együttműködésével írjuk le. Ez azt jelenti, hogy a hívás osztáson belül funkció osztást is végzünk. Egy hívásnak lehet külön regiszter, összekötő stb. funkciókat leíró automatája.

A számítógépes realizáció általában hűen követi a fentieket. Minden keletkező híváshoz egy jobot rendelünk, mely a hívás végén fejezi be tevékenységét. A rendszeren belül egyidejűleg sok job lehet aktív, melyek egymástól függetlenül különböző állapotokban lehetnek. A jobok a programtárat közösen használják (minden hívást ugyanaz az absztrakt automata írja le), csak a saját (a hívást jellemző) pillanatnyi adataik számára kell külön, más job által hozzá nem férhető operatív memóriarészt biztosítani. Így a rendelkezésre álló memória elvileg megszabja, hogy egyidejűleg hány job lehet aktív, azaz hány parallel hívást tud a rendszer feldolgozni. Az előbbiekben említett funkció-osztást a konkrét megoldások során úgy realizáljuk, hogy az egy jobhoz tartozó operatív memóriarészt több, funkcionálisan elkülöníthető részre osztva ezekhez külön automatákat (állapotokat) rendelünk.

2. A QA96 hívásfeldolgozó automatái

A QA96-ban a hívások pillanatnyi állapotát jellemző adatokat két jól elkülöníthető csoportban tároljuk. Az egyes vonalak, ívpontok (mellékállomás, trunk, kezelő) néhány jellemző állapotának tárolására szolgáló, az illető ívpontokhoz fixen hozzárendelt tárolóelemek képezik az egyik, és a vonalakhoz a hívás idejére hozzárendelhető memóriaterületekből (hívástárak) összeálló közös memória képezi a másik csoportot. Az előbbi, a vonali „erőforrások" állapotátmeneteit a vonali-automata, a másik erőforrás, a hívástár működését nagyrészt a hívásautomata írja le, mint ahogy azt az alább közölt SDL diagramokon láthatjuk.

A hívások lebonyolításában tehát az említett két erőforrás vesz részt, melyeket a hívás idejére egymásra mutató pointerek rendelnek össze. A hívás során a köztük levő kétirányú logikai kapcsolat a központ belső jelzésrendszerén keresztül jön létre. A jelzések az átvitel irányának megfelelően a bemutatott diagramokon mint kimeneti, ill. bemeneti jelzések szerepelnek. Az SDL nyelv további szabályai szerint szerepelnek belső jelzések is (többnyire időzítések), me-



2. ábra. A „vonali" automata

lyek az adott állapothoz rendelt és onnan indítható tevékenységek eredményeképpen születnek.

A vonali automata *(2. ábra)* feladata a vonali jelek feldolgozása és továbbítása a hívástár automatája felé. A vonal szabad állapotát két esemény változtatja meg: híváskezdeményezés (hurok záródik) vagy egy a vonal felé irányuló hívás (hívás érkezik). Híváskezdeményezéskor az első feladat egy szabad hívástár (HT) kijelölése, és a hívásautomata indítása.

A vonal FOGLALT állapotba, vagyis definíciószerűen abba az állapotba, amikor a további vonali jelek feldolgozása indulhat, a hívástár aktivizálódása után kerül, amit többek közt az említett pointerek beállítása jellemez. Ezután a vonali automata a hurokváltozások számát közvetíti a hívástárba, amiből kiértékelhető a betárcsázott számjegy, és a vonal állapota a változás után (a számláló páros vagy páratlan értéke). Ha az első változás a hívástár aktivizálása után a hurok $\varnothing \rightarrow 1$ átmeneténél következik be, akkor ez csak csengetéskor a hívott jelentkezését jelentheti.

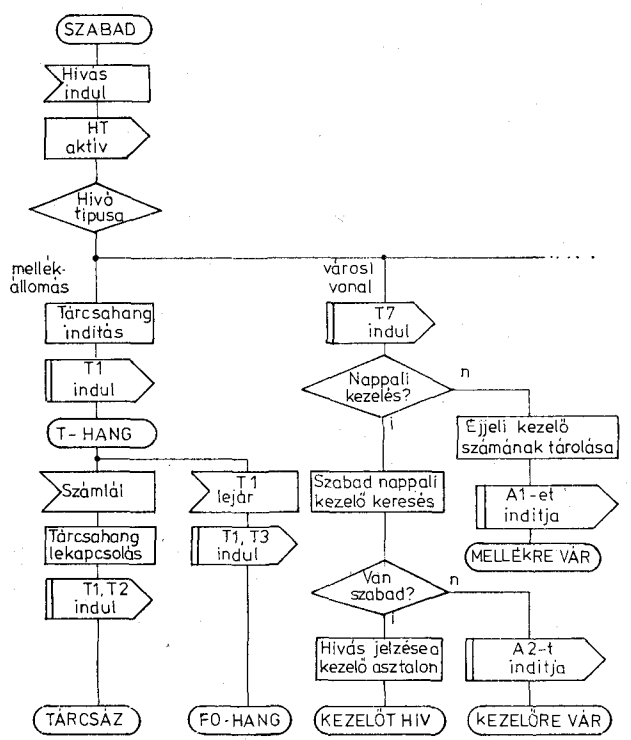
A vonali automata a „hívás indul" jelzésben közli a hívó vonal adatait, melyek az ívpontra jellemzőek. Ebből, a központ áramköreit leíró „adatbázis" (lásd 4. pont) segítségével a hívástár automatája határozza meg a hívó típusát (mellékállomás, trunk, LB vonal, tie-line stb.) és ennek alapján az első állapotátmenetet *(3. ábra)*. Mellékállomás esetén az automata T-HANG állapotba kerül, melyből az első hurokváltozás vagy az első állapotátmenet során indított T1 időzítés lejárta mozdítja ki. A hurokváltozásokat közvetítő jelzések vételekor indított T2 időzítés az impul-

zus-sorozat végének megállapítására, ill. a bontás késleltetésére szolgál.

Az első számjegy vétele után szükség van annak elemzésére. Tekintve, hogy számjegyanalízisre a hívásfeldolgozás során számtalan más helyen is szükség van, célszerű ezt egy külön „automatával" megvalósítani, mely az analízis eredményeit jelek formájában közli a hívásautomatával *(4. ábra)*. Programozók ilyenkor egyszerűen „eljáráshívásról" beszélnének. Valóban, ezt az SDL nyelven ilyen módon lehet ábrázolni.

A QA96-ban realizált programstruktúra mégis inkább az „analízis-automata" megoldást tükrözi, tekintve, hogy a közvetlen eljáráshívást a futási idők korlátozása céljából komplikáltabb feladatok esetén nem alkalmaztuk. Így az analízis-automata révén a számanalízissel és a hívásautomata szóban forgó állapotaiban végzett kapcsolódó tevékenységekkel eltöltött futási idő nem egyszerre terheli a processzort, hanem időben elosztva. Ez a forgalmi csúcsok elviselésére kedvezőbb helyzetet teremt. Ugyanezek vonatkoznak a linkkeresés automatájára is, ami valóságosabb automata, hiszen ennek legalább van két állapota, ami a feladat további osztásából adódott (keresés és a kiválasztott linkek bejegyzése a linktérképbe, lásd *5. ábra*).

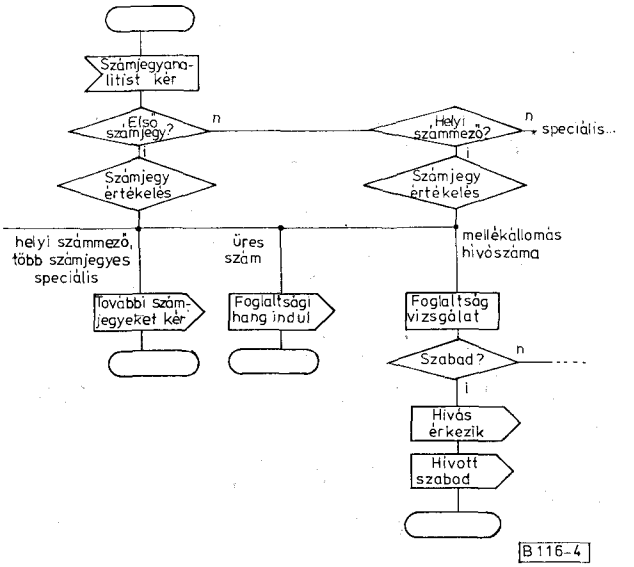
Az idézett automaták kimenő jeleire a hívásautomaták valamelyik VÁR állapotban várakoznak *(6. és 7. ábrák)*. Helyi hívás esetén a kapcsolóút le-



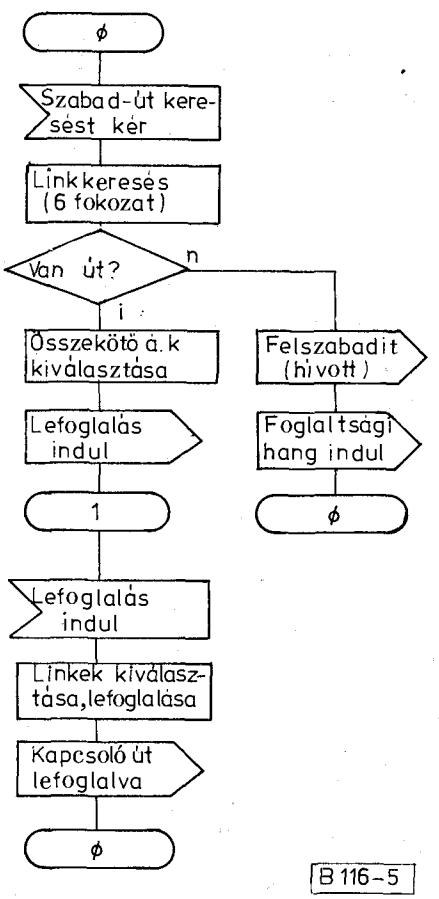
T1: regiszter időzítés (~ 15sec)
T2: bontás késleltetés (~ 0,2sec)
T3: foglaltsági hang félperiódusa (~0,3sec)
T7: fővonali időzítés (~40sec)
A1: mellékállomás felszabadulását jelző akció
A2: kezelő szabaddá válását jelző akció

B116-3

3. ábra. Hívástár állapotátmenetei

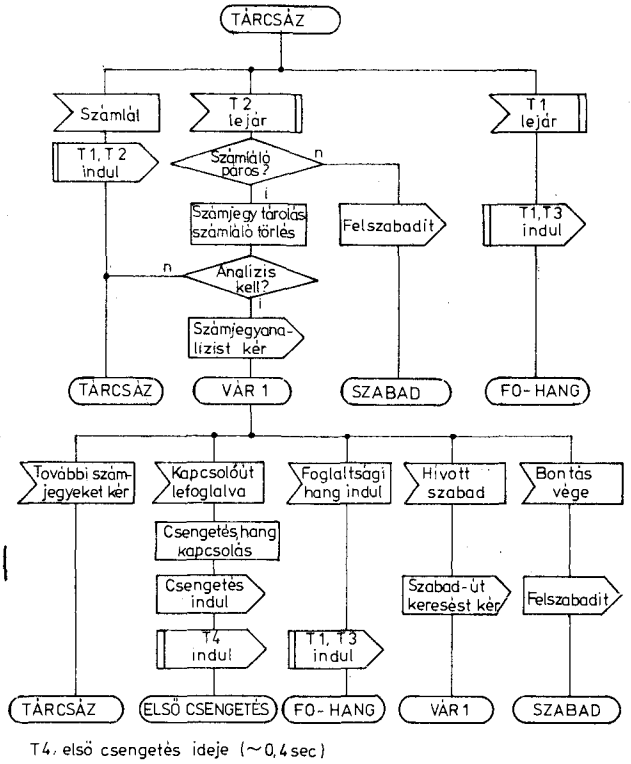


4. ábra. Számjegyanalízis



5. ábra. Linkkeresés

foglalása után a hívásautomata T4 ideig tartózkodik az ELSŐ CSENGETÉS *(8. ábra)* állapotában, majd a periodikus csengetést és csengetési hangot vezérlő (itt nem ábrázolt) belső akció indítása után a CSENGET állapotban várja meg a hívott jelentkezését. Ekkor indul a kapcsolófokozatok működtetését vezérlő automata *(9. ábra)*, melynek három állapota



6. ábra. **Hívástár állapotátmeneteí**

a hardware által megszabott, T6 idejű működési fázisoknak felel meg. A sikeres kapcsolást követi a BESZÉD állapot *(7. ábra)*. Végül a bontás különböző feladatait is külön automata végzi, melyek befejezése után kerül sor a hívástár és a vonalak felszabadítására, illetve a bontott fél felé a foglaltsági hang indítására.

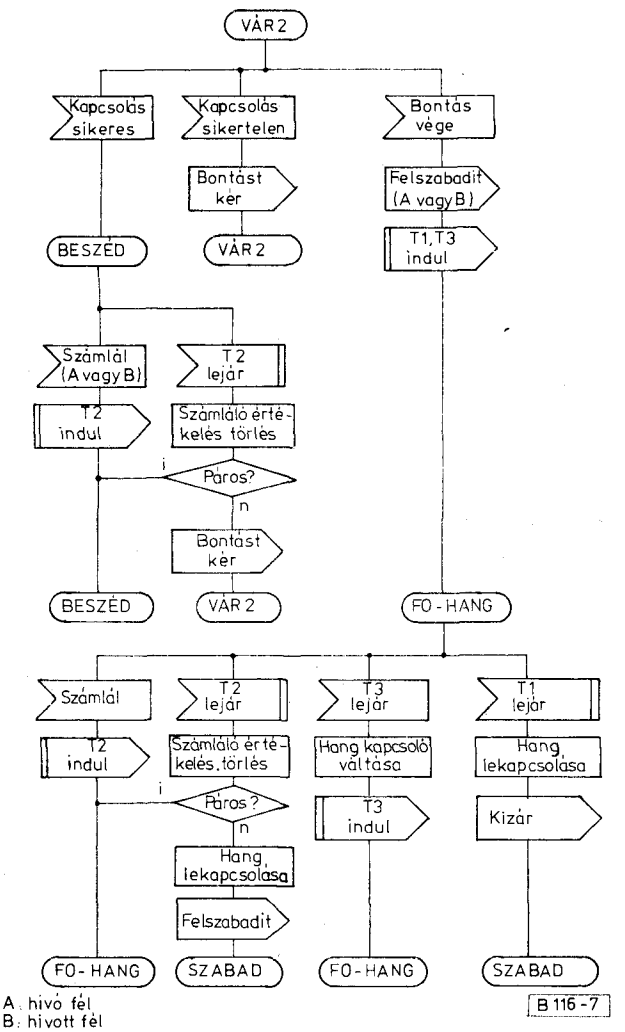
3. A programok szervezése

A tárgyalt automaták állapotátmeneteit realizáló programokon kívül a különböző perifériák adatbeviteli (letapogató) és kiviteli programjai teszik teljessé a hívásfeldolgozó rendszert. Az ütemezésre a lehető legegyszerűbb módszert választottuk: a programokat prioritás szerint fixen egy táblázatba rendeztük, és végrehajtásuk e szerint ciklikusan egymást követi („round robin"). Ezt a sorrendet csak az időmegszakítási rendszer bonthatja meg, a 20 ms-os ciklusidő leteltekor mindig a táblázat elejéről indul a feldolgozás. A táblázat utolsó programja arról gondoskodik, hogy az említett 20 ms-os ciklusidőből még hátralevő időt eltöltse. E szerint a lista programjai nagyjából ütemesen kerülnek behívásra, a futási időkből adódó pillanatnyi eltérések hosszabb időtartamra nézve kiegyenlítődnek.

A hívástárak állapotmeneteit kezelő program szervezése olyan, hogy egy adott hívástárral a központ 100 ms-onként foglalkozik, függetlenül attól, hogy az aktív-e vagy sem. A rendszer hátránya, hogy az inaktív hívástárak (vagyis amelyek éppen nincsenek „állapotátmeneti" fázisban) is vesznek el némi futási időt, viszont nagy előnye az egyszerűségen kívül, hogy az időzítések magával a programhívások számával mérhetők, ahol egy egység 100 ms-ot jelent. Ezt az „időosztási" elvet más erőforrásra nézve is alkalmaztuk (pl. a blokkolt vonalak felügyeletét végző program egy behívás alkalmával csak nyolc vonallal foglalkozik, így az elvileg lehetséges max. $512=8\times64$ vonallal $64\times20=1280$ ms alatt végez. Ez az az idő, ami alatt a központ egy blokkolt vonal felszabadulását regisztrálni köteles).

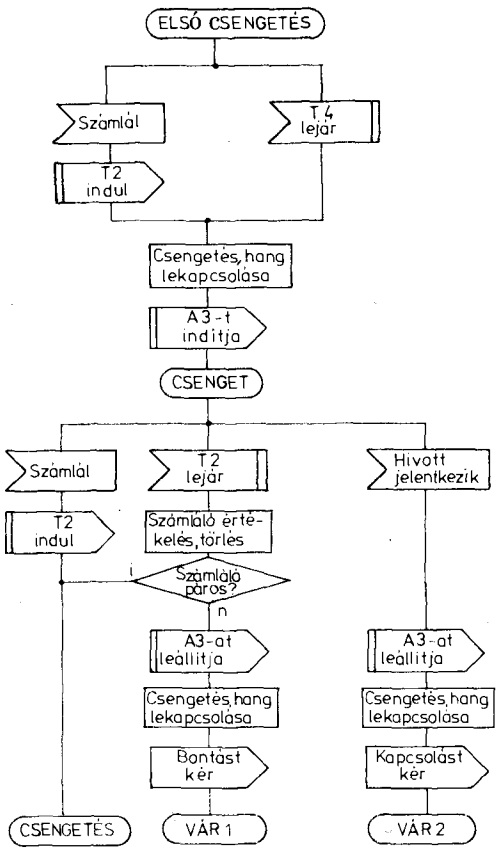
A vonalak letapogatása vagyis állapotuk leolvasása, a memóriában nyilvántartott állapotukkal való összevetése és a változásoknak megfelelő „üzenetek" feljegyzése a memória adott helyeire természetesen minden rendszer ciklusban megtörténik. Feldolgozásuk, vagyis az automaták állapotátmeneteinek végrehajtása történik más-más időben, a feladat jellegétől függően. Pl. a vonali automata „számlál" jelzése a vonali erőforráshoz rendelt hívástár adott rekeszében realizálódik, melyet a hívástár feldolgozó program, ha szükséges, 100 ms-onként értékel.

Ugyanígy a többi letapogató program (billentyűs készülékek hangfrekvenciás vevői, kezelői billentyűzetek, szerviztáska billentyűzete) is 20 ms-onként



7. ábra. **Hlvástár állapotátmenetei**

vesz mintát a „külvilágból" és állit elő „jelzéseket" a többi feldolgozó programnak. A feldolgozó programok egymásnak küldött jelzéseivel együtt ezek a jelzések adják az említett belső jelzésrendszer jelkészletét. A jelzések átvitele adott memóriacellák feltöltésével és lekérdezésével történik.



A3 Normál periodikus csengetést és csengetési hangot előállító akció.

B 116-8

8. ábra. Hívástár állapotátmenetei

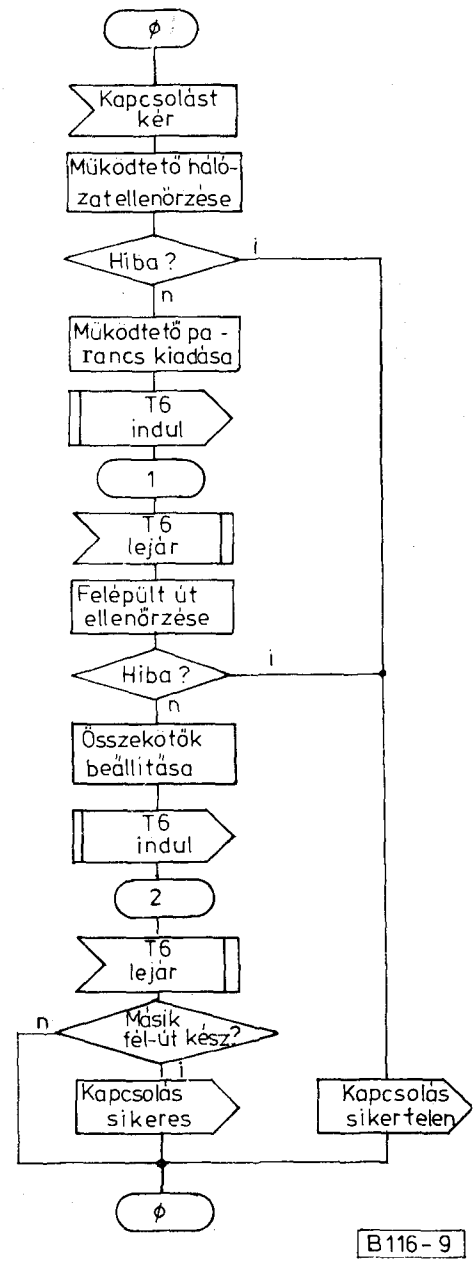
4. Az adatok szervezése

Az adatok gondos strukturálása az automaták tervezésével egyenrangú, igen fontos feladat. Adatok alatt az eddig szereplő „erőforrásokat", valamint egy adott központ konkrét kiépítését leíró„ adatbázist" értjük. Ezek logikai szervezése, a memóriában való olyan elrendezése, hogy az a feldolgozó programok számára egyszerű algoritmusok használatát tegye lehetővé, igen lényeges (pl. a futási idők alakulása szempontjából). Már a tervezés korai stádiumában tudni kell, hogy melyek azok az adatok, melyek lehetőleg direkt módon rendelkezésre kell álljanak, mert gyakran van rájuk szükség, és melyek a kiszámíthatóak. Sokszor érdemes redundanciát alkalmazni, hiszen „amit vesztünk a réven, megtérül a vámon" vagyis az egyszerűbb algoritmusok a programtár méretének csökkentését is jelentik, így a memóriaszükséglet végső soron változatlan maradhat.

A QA96 adatbázisa lényegében a következő adatokat tartalmazza:

- a kapcsolómező (linkbekötés) kiépítési adatai,
- az áramkörök és azok hívószáma az ívpontok (helyszám) szerint rendezve,
- az áramkörök helyszáma hívószám szerint (4 db 100-as mező) rendezve,
- a mellékállomási kategóriák hívószám szerint rendezve,
- vonalnyalábokba tartozó áramkörök, nyalábonként felsorolva (trunkök, PBX csoportok, egyéb áramkörök),
- hívószámtranzláció táblázatai,
- az alközponti kezelőkre vonatkozó adatok stb.

Mindezek az adott esetben egy 2 kbyte-os területen helyezkednek el REPROM tárolókban rögzítve. A központ rendelésénél, a rendelésfelvételi adatlapok adatainak gépi rögzítése után automatikus rendszer



B116-9

9. ábra. Kapcsolóút működtetés

generálja az említett tokok programozásához szükséges lyukszalagot és minden szükséges dokumentációt.

A hívástárak két ívpont adatainak tárolására képesek, méretük egyenként 31 byte. Ezek részben fix rendeltetésűek, adott funkciójuk van, mások az állapotoktól függően többszörösen ki vannak használva. Az ívpontokhoz egyénileg hozzárendelt tárolók ívpontonként ~2 byte-ot tesznek ki. A különleges szolgáltatások (hívásátirányítás, rövidített hívószámok stb.) mintegy 2 kbyte-ot vesznek igénybe. A programrendszer működéséhez szükséges változó adattár mérete ~7 kbyte.

5. Eredmények, tapasztalatok

A programozás assembly nyelven történt, a fordítást, listázást cross-assembler és szerkesztő segíti. A programtár mérete 20 kbyte-ra adódott. A programok a fejlesztés és az üzemi próbák során igen sokat változtak, a jelenleg „gyártott" program (REPROM, ill. PROM változatban) az elsőnek kibocsátott rendszer teljesen átdolgozott változata, és az újabb és újabb üzembe állítások még ma is szolgáltatnak „meglepetéseket" (az anyag írásakor mintegy 10 000 vonal QA96 gyártása fejeződött be). A legtöbb probléma a kezelő-készlet billentyűzetének „abnormális" használatából adódott. Általában a „nem betervezett" eseményekkel, tranziensekkel van a baj, amit csak igen gondos tervezéssel lehet megelőzni és hosszú kimerítő teszt kell, hogy megelőzzön minden újabb fejlesztési produktum vagy javítás tényleges üzembe állítását.

A gondos tervezés, a tervezési módszerek állandó javítását is kell jelentse. Bebizonyosodott, hogy az előkészítő tervezési fázisban (rendszeranalízis, specifikációk) jóval több idő és energia befektetése szükséges, mint hittük, az ebben a szakaszban elkövetett hiba sokkal nehezebben javítható, mint pl. a kódolási hibák. Ki kellett alakítani a specifikációk és rendszertervek készítésének gyakorlati módszereit, dokumentálási formáit. Például az SDL nyelvet, mint tervezési segédeszközt a QA96 tervezésénél még nem használtuk, a bemutatott ábrák utólag születtek.

Real-time rendszerek tervezésénél igen lényeges, hogy még a tervezés korai stádiumában becsülni tudjuk a software forgalmi terhelhetőségét. Ehhez objektív módszerek is szükségesek, és esetünkben a számítógépes szimuláció adott egy ilyen módszert. A tervezett software vázát vagyis az egyes programok strukturális összefüggéseit, a belső jelzésrendszert képeztük le, a jelzések hatására végzendő tevékenységeket az ahhoz szükséges rutinok becsült futási idejével „helyettesítettük". Ezt az absztrakt rendszert a külső jeleknek egy másik program által keltett mesterséges „forgalmával" hajtottuk meg, és értékeltük ennek különböző intenzitása mellett a fellépő válaszidőket, várakozási sorok hosszát, foglalt erőforrások számát stb. A vizsgálat módszerei külön cikket érdemelnek, itt csak annyit említünk, hogy a rendszert a QA96 software tervezésénél már használtuk, és az utólagos ellenőrzés a software forgalmi teherbíró képességét a tényleges adatok birtokában is megfelelőnek találta. Ezt egyébként az eddigi installációk is alátámasztják.

Köszönetnyilvánítás

Szeretnék köszönetet mondani minden munkatársamnak, akik a munkában résztvettek és értékes tanácsaikkal, ötleteikkel és nem utolsó sorban kitartásukkal a munka sikerét biztosították. Ugyancsak köszönetet mondok a BHG Fejlesztési Intézet vezetőinek támogatásáért.

IRODALOM

[1] *Kawashima, H.:* Functional specification of call processing by state-transition diagram. IEEE Trans. 1971. COM—19 p. 581—587.

[2] *M. T. Hills, S. Kano:* Programming electronic switching systems. 1976. Published by Peter Peregrinus Ltd.

[3] *Däcker, I. Jacobson:* Real time system design using CHILL. Conference Publications, IEE Software Engineering for Telecommunication Switching Systems, 1978. Helsinki.

[4] *R. T. Boute:* Logical models for computer control of telephone exchanges. Conferenee Publications, IEE Software Engineering for Telecommunication Switching Systems, 1978. Helsinki.

[5] CCITT ajánlások; Orange Book, Vol. VI/4 Z101—Z104.