AZ IPARI ETHERNET, MINT VALÓS IDEJŰ RENDSZER

Dr. Ferenczi István, villamosmérnök, Nyíregyházi Egyetem,
e-mail: ferenczi.istvan@nye.hu

1. A valós idejű rendszerekkel kapcsolatos alapfogalmak

A valós idejű rendszerek fogalma a digitális számítógépek automata célgépekként történő alkalmazásával egy időben jelent meg. A mikroszámítógépek és mikrokontrollerek egyre inkább beépülnek a vezérléstechnikai rendszerekbe. Egyre könnyebb programozásuk és egyre kedvezőbb áruk miatt, már a legegyszerűbb automaták is tartalmaznak valamilyen mikroprocesszort, vagy mikrokontrollert. A valós idejű rendszerekben az események feldolgozásában az idő alapvető szerepet játszik. Az alapprobléma, hogy egy vagy több külső eszköz előre nem ismert időpontban valamilyen esemény által kiváltott információt küld a számítógépnek, amelyre az eseményt kiváltó októl függő, előírt időn belüli választ kell küldeni. A környezetből egyidejűleg több eseményhez kapcsolódó üzenet is érkezhet a számítógéphez és ebben az esetben is teljesülnie kell az előírt időn belüli válaszadásnak.

1.1. A valós idejű viselkedés meghatározása

Az ilyen rendszerek sajátossága, hogy a fizikai környezetükkel aktív, valós idejű információs kapcsolatban állnak. A rendszer bemenetére érkező információkat kiértékelik, feldolgozzák és meghatározott időn belül kell választ adniuk. Ezeket a rendszereket a befogadó természetes környezet és a mesterségesen létrehozott hardver és szoftver komponensek nagymértékű összekapcsolódása, együttműködése jellemzi és a környezetükkel, – amely jellemzően nem informatikai rendszer – közvetlen információs kapcsolatban állnak. Ezek, általában olyan digitális eszközök, amelyek speciális célra készültek, és nem rendelkeznek bonyolult felhasználói felülettel.

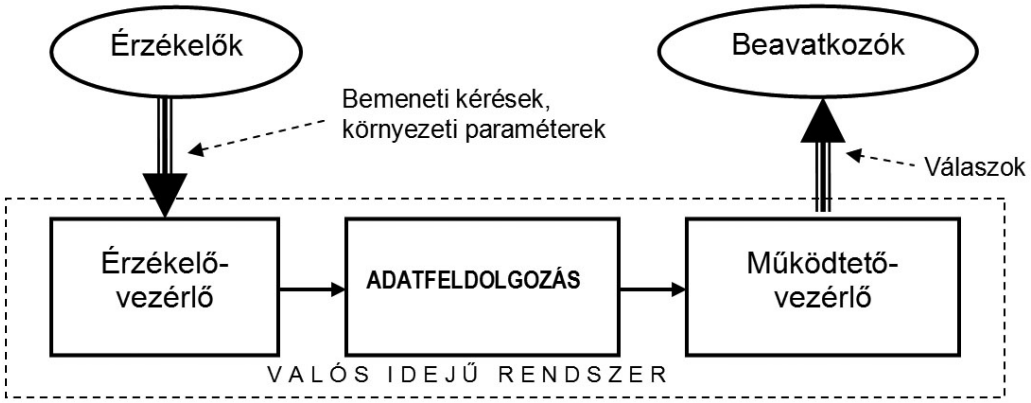
A valós idejű rendszerek definícióját csak a tulajdonságai alapján nem tudjuk meghatározni. Egy meglévő rendszerről addig nem lehet eldönteni, hogy valósidejű-e vagy nem, amíg nem ismerjük meg a rendszer feladatspecifikációját. Bizonyos időhöz kötött tulajdonságok alapján feltételezhetjük, főleg hogyha időmérő eszközöket, vagy időfüggő algoritmusokat használ. A feladat ismerete nélkül azonban nem dönthető el, hogy ezek használata valamilyen időbeli viselkedésre vonatkozó követelmény teljesítéséhez vagy más szempontok miatt szükséges.

A definíciót tehát nem a rendszer tulajdonságai, hanem a vele szemben támasztott követelmények alapján kell megadni. Az évek során számos klasszikus értelemben vett definíció látott napvilágot [1, 2, 3], melyek alapján kijelenthető: *Valós idejűnek tekinthetjük azt a rendszert, amelyek specifikációjában a logikai és számítási funkciókon túlmenően valamilyen időbeli viselkedésre vonatkozó előírás is szerepel a külső, valós időskálához kötötten [4]*.

1.2. A valós idejű rendszerek fontosabb jellemzői

Egy valós idejű rendszer lényegét tekintve abban különbözik egy adatfeldolgozó rendszertől, hogy:

- a környezetéből érkező jeleket érzékelők szolgáltatják. Ezeknek feladata a működtetett (irányított) rendszer, környezet állapotáról információkat adni.
- a feldolgozás eredményét (választ) előírt idő alatt kell a beavatkozó eszközre juttatni. (1. ábra)



1. ábra. Egy valós idejű rendszer általános modellje [4]

A feldolgozó program nagysága természetesen az általa megvalósítandó feladatot leíró algoritmustól függ, de funkcionálisan tartalmazza az 1. ábra szerinti három rész funkciót is, amelynek alapján folyamat jól meghatározható. Ez szekvenciális programokban nem mindig kivitelezhető. Ezért a valós idejű rendszereket konkurens, együttműködő folyamtokként szokás tervezni. Ezeket a folyamatokat a valós idejű rendszer specifikusan kialakított operációs rendszere kezeli. A folyamatok futtatása rendszerint ciklikus, ebből következik, hogy egyik alapvető időtényezője a rendszernek éppen a *ciklusidő* nagysága lesz.

Egy valós idejű rendszer fontosabb jellemzői a következők:

- az irántyó berendezés rendszerint valamilyen intelligens rendszer (PLC, mikrokontroller, de akár PC is lehet),
- idő és/vagy esemény vezérelt működés,
- az irányító rendszer online kapcsolatban van a technológiai berendezéssel az érzékelő, beavatkozó eszközökön keresztül,
- az előírt *időkorlát* (határidő, deadline) az irányított technológiai berendezés, folyamat időviszonyaiból határozható meg.

A számítógépek ilyenfajta alkalmazása szükségessé tette, hogy a számítógépes rendszer és a benne futó programok időbeli viselkedését is vizsgáljuk, illetve specifikáljuk, mégpedig a környezetében érvényes, valós időskálán. A számítógépes rendszernek úgy kell működnie, hogy *válaszideje* (response time) a megengedett időkorláton belül maradjon. Ha nem sikerül adott időn belül reagálni, az ugyanolyan hiba, mint egy téves számítási eredmény előállítása, sőt következményeit tekintve bizonyos területeken (pl. ipari biztonsági rendszereknél) még súlyosabb is lehet.

A valós idejű rendszerek általában két csoportra oszthatók:

- **szigorúan valós idejű (hard real-time)** rendszerek, amelyek specifikációja egyértelműen rendszerhibának tekinti valamely időkövetelmény be nem tartását. Ez azt jelenti, hogy a szigorúan valós idejű rendszerek normál működése során nem

engedhető meg, hogy valamilyen időkövetelmény teljesítése elmaradjon. Ha mégis bekövetkezne ilyen esemény, a rendszernek a kidolgozott stratégiának megfelelően kell reagálnia, semmiképp sem szabad határozatlan állapotba kerülnie.

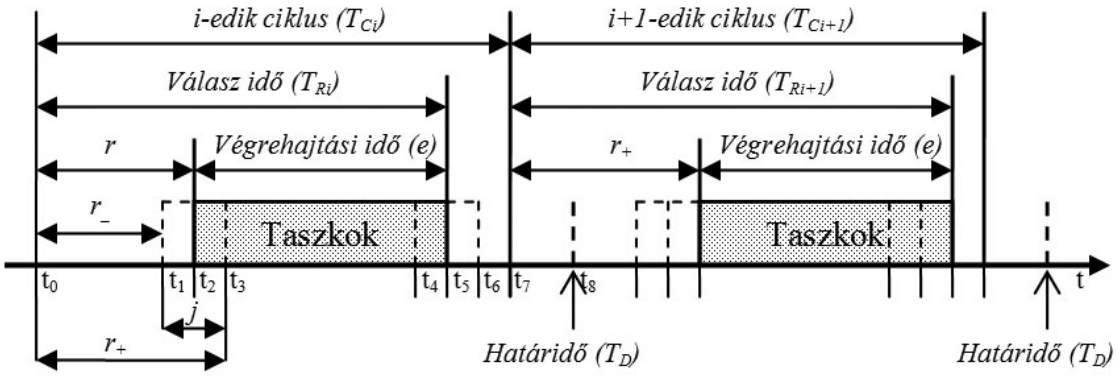
- **lazán valós idejű (soft real-time)** rendszerek, amelyek körében az ilyen mulasztást csak a rendszer működőképességének vagy teljesítményének csökkeneséként fogjuk fel, és erre az esetre is specifikáljuk a rendszer viselkedését.

1.3. Egy valós idejű rendszert jellemző fontosabb időtényezők

A valós idejű rendszerek elemzésekor számos időtényezőt kell figyelembe vennünk, azért, hogy optimalizálni tudjuk a működést, azaz minimálisra csökkentsük azokat a tényezőket, amelyek veszélyeztethetik a valósidejű működést. Ezek közül a legfontosabb időintervallumok a következők:

1. előkészületi idő (release time),
2. végrehajtási idő (execution time),
3. válasz idő (response time),
4. ciklus idő (cycle time),
5. remegési intervallum (jitter),
6. határidő (deadline).

Ezeknek az időknek az egymáshoz való viszonyát mutatja a következő, 2. ábra.



2. ábra. A valós idejű működéssel kapcsolatos időtényezők

Az *előkészületi idő (r)* meglehetősen tág fogalom. Vonatkozhat valamilyen hardver eszközre, amikor azt az időtartamot jelenti, amíg az eszköz elérhetővé vagy használhatóvá válik. De vonatkozhat valamilyen feldolgozásra szánt adatra is, amelyet valamelyik buszról kell levenni. A *végrehajtási idő (e)*, egy adott feladat teljes feldolgozásához szükséges idő. A *válaszidő* az eseményt (feladatot) kiváltó időpillanattól a beavatkozás megkezdéséig eltelt időt jeleni. A *ciklusidő (T_C)* a ciklikus működésű rendszerek legfontosabb jellemzője. Azt az időintervallumot jelenti, amelynek eltelte után a taszkok végrehajtása ismétlődik. A *jitter (j)* rendszerint azt az időintervallumot jelenti, amelyen belül várható valamely fentebb felsorolt tényező kezdési időpillanata. Sok esetben nem tudjuk pontosan meghatározni, vagy bizonyos okok miatt nem mindig azonos például az előkészületi idő esetében. Azt viszont meghatározhatjuk, hogy egy [r_-, r_+] intervallumon belül legyen. Maximális értéke:

$$j_{max} = r_+ - r_- \tag{1}$$

A jitternek főleg a szinkronizált időkapcsolatok esetében van jelentősége.

A *határidő (T_D)* meghatározza azt a maximális időtartamot, amely alatt a rendszernek be kell fejeznie a feladatok végrehajtását (2. ábra). Ezt rendszerint a külső, technológiai követelmények határozzák meg [5].

A fentebb felsorolt időtényezők korántsem mondhatóak állandónak. Bármelyik változhat ezek közül, akár egyszerre is, de a lényeg az, hogy ne lépje túl a határidőt. Ciklikus működésű rendszereknél a bemeneti értékek beolvasása, és a kimeneti csatornák írása is a ciklikusan történik. A legtöbb esetben ezek az események részei az alapciklusnak. Logikusnak tűnik, hogy a bemeneti változók beolvasása a ciklus első fázisában, még a taszkok elindítása előtt megtörténjen, majd a ciklus végén a válaszok megjelenjenek a kimeneteken. Ez a legtöbb rendszernél így is van, de létezik olyan megoldás is, amikor mindkét esemény a ciklus elején megtörténik, sőt ezen belül, a kimenetek írása megelőzi a bemenetek beolvasását.

Az imént említett perifériakezelési stratégia miatt előfordulhat, hogy valamely bementi gerjesztés lekési a bemeneti aktualizálási fázist, így csak a következő ciklusban kerül kiszolgálásra. (Feltételezzük, hogy a gerjesztés megfelelően hosszú időtartamú.) Ezért a legkedvezőtlenebb esethez (worst case) hozzárendelt maximális válaszidő (T_{Rmax}), akár a ciklusidő néhányszorosa is lehet. Ennek következtében a válaszidők mindig nagyobbak, legfeljebb egyenlők lehetnek a ciklusidővel, de mindig a technológiai határidőn belül kell megvalósuljanak. Következik, hogy:

$$n \cdot T_C \le T_{R\,max} \le T_D ; \quad n = 1, 2, \ldots \tag{2}$$

1.4. Az ipari valós idejű rendszerekkel szemben támasztott egyéb követelmények

Az ipari környezetben alkalmazott valós idejű rendszereknél az is meghatározó szempont lehet, hogy milyen módon tudja kezelni az esetleges időkorlátokra vonatkozó hibákat. Az semmiképp sem megengedett, hogy egy kritikusnak számító időhiba miatt a rendszer csak egy hibaüzenettel reagáljon, és várakozó állapotba álljon, amíg valamilyen operátori beavatkozás nem történik. A rendszertől elvárt viselkedés általában az, hogy próbálja meg bizonyos mértékig csökkenteni a hiba (például határidő elmulasztása) következményeit, és később automatikusan térjen vissza a normál üzemállapotba, vagy ha ez sikertelen, akkor minimum elvárás, hogy legalább a biztonsági funkciókat helyezze működésbe.

Az ilyen rendszerekkel szemben tehát *fokozott biztonsági és megbízhatósági követelményeket* támasztanak. A fokozott biztonság érdekében a rendszer akkor sem adhat ki olyan kimenőjeleket, amelyek a környezetben veszélyes helyzetet idéznek elő, ha meghibásodás miatt nem tudja folytatni működését. A fokozott megbízhatóság elérésének érdekében a rendszereket nemcsak a biztonsági funkciókkal, hanem a *hibatűrés* képességével is fel kell ruházni.

Ugyancsak a biztonsággal és a megbízhatósággal kapcsolatos követelmény a *robosztusság.* Egy robosztus rendszert különlegesen kedvezőtlen, szélsőséges környezeti hatások sem tehetnek tönkre. Átmenetileg leállhat, de a kedvezőtlen hatások elmúltával tovább kell működnie. Egy ilyen a rendszer akkor sem kerülhet határozatlan állapotba, ha nem várt, nem specifikált vezérlőjeleket kap, vagy a program végrehajtása valamilyen meghibásodás, tápkimaradás miatt megszakad, majd újraindul. Az esetek jelentős részében még kezelői segítségre sem lehet számítani, hanem *automatikus újraindítást (watchdog),* esetleg *alapkonfiguráció* betöltést kell megvalósítania.

A valós idejű vezérlési rendszerek gyakran hónapokig, sőt évekig nem állíthatók le, vagyis gyakorta *folyamatos működésűek*. Például egy folyamatosan működő technológiát irányító rendszerénél üzemszerű működés mellett lehet, hogy egy évben csak egyetlen egy

tervezett karbantartási leállás engedhető meg, amelynek időtartama néhány nap, legfeljebb egy hét lehet. Minden egyéb leállás hatalmas anyagi veszteséget vagy biztonsági kockázatot idézhet elő.

Ugyancsak gyakori, hogy egyes rendszerek vagy a nagyobb rendszerek alrendszerei, hosszú időn át felügyelet nélkül működnek. Ilyen *felügyelet nélküli működést* kell biztosítani például a terepre kihelyezett irányító, adatgyűjtő állomásokra, vagy például a műholdak, űrszondák esetében, ahol bármilyen helyi kezelői beavatkozás, diagnosztika, javítás, újraindítás nem lehetséges.

Az eddig felsorolt tulajdonságok miatt a valós idejű rendszerek szoftverei sokkal nagyobb része foglalkozik a talán soha be nem következő, kivételek kezelésével, mint egyéb rendszerekben. Ez még nehezebbé teszi az időkövetelmények teljesítését.

A valós idejű rendszerek feladatainak megfogalmazása leggyakrabban a rendszer viselkedésének leírásával történik. A leírás azt tartalmazza, hogy egy-egy funkció végrehajtása közben milyen üzenetváltások történnek a környezeti szereplők és a rendszer között, és ezek hatására milyen műveleteket kell végrehajtani. Az irányítás tárgyát képező aktív környezeti folyamatok általában egymástól függetlenül, időbeli korlátozások nélkül kezdeményezhetnek beavatkozásokat. A rendszer ennek következtében egyidejűleg több környezeti szereplővel folytathat párbeszédet úgy, hogy a különböző szereplőkkel váltott üzenetek egymáshoz viszonyított sorrendje előre nem határozható meg [4].

2. Valós idejű alaprendszer

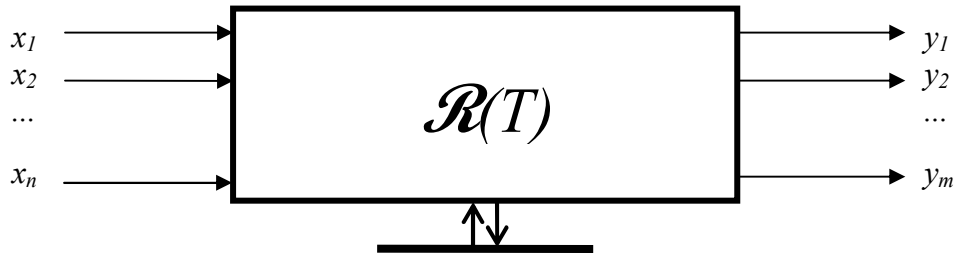
A valós idejű Ethernet rendszerek elemzésénél célszerű kiindulni egy olyan alaprendszerből, amely minden szempontból eleget tesz a valós idejű alapkövetelményeknek. Első sorban ciklikus működésű rendszert feltételezünk, amelyben a feladatok (taszkok) futása ütemezhető, bizonyos feladatok pedig akár párhuzamosan is végezhetőek. A valós idejű rendszer időbeli viselkedésének leírásában a leggyakoribb időhöz kötött feladatok a következők:

- *határidős feladat:* adott időn belül végrehajtandó,
- *periodikus feladat:* adott időközönként ismételten végrehajtandó, (legtöbbször határidős ez is)
- *prioritással kitüntetett feladat:* a prioritási szinttől függően akár megszakítást is kezdeményezhetnek a határidős feladatokon belül is (pl. alarmok),
- *időzített feladat:* bizonyos megengedett eltéréssel egy adott időpontban végrehajtandó,
- *időkorlátos várakozás:* a rendszer külső esemény bekövetkezésekor végez el egy feladatot, de ha a külső esemény nem következik be adott időpont eléréséig, más feladatot kell végrehajtani.

2.1. Alapvető tényező a ciklusidő

Ahhoz, hogy az előbbiekben felsorolt idő-specifikus feladatokat a rendszer maradéktalanul képes legyen megvalósítani, alapműködése is valamilyen formában időhöz kötött kell legyen. Itt rendelhető hozzá a már az előzőekben ismertetett *ciklusidő*, amely egy alaprendszer esetében nem más, mint a felhasználói program és a hozzá kapcsolódó rendszerfeladatok ismétlési gyakoriságának ideje. Az említett feladatok közül, a periodikus és határidős feladatok végrehajtása a legkritikusabb, ezért e két szempont szerint fogom meghatározni az alaprendszer kritériumait. Feltételezzük a következő ábra szerinti alaprendszert, ahol $x_1, x_2, \ldots, x_n$ a rendszer bemeneti, $y_1, y_2, \ldots, y_m$, pedig a kimeneti, véges

számú változói, $\mathcal{R}(T)$ a közöttük fennálló relációk halmaza és a rendszer rendelkezik Ethernet csatlakozással.



3. ábra. Valós idejű alaprendszer

<u>Megjegyzés:</u> Az esetek zömében $n \geq m$, mert egy-egy kimeneti eseményt rendszerint több bemeneti feltétel teljesülése határoz meg.

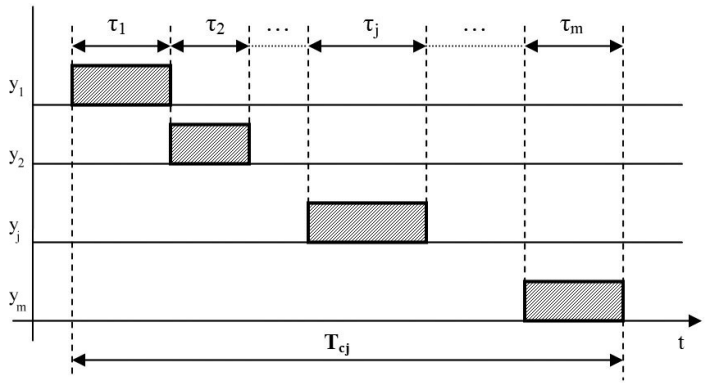
A kimeneti eseményeket meghatározó relációk ($\mathcal{R}_j$, $j = 1,2,...,m$) a bemeneti események közötti logikai kapcsolatoktól (f_j), az általuk kiváltott folyamatoktól, valamint az ezeket meghatározó időktől függenek. Általános formában felírhatjuk a következőket :

$$
\begin{aligned}
&y_1 = \mathcal{R}_1[f_1(x_1,x_2,...x_i,...,x_n),\ T_{R1},\ D_1] \\
&y_2 = \mathcal{R}_2[f_2(x_1,x_2,...x_i,...,x_n),\ T_{R2},\ D_2,] \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&y_j = \mathcal{R}_j[f_j(x_1,x_2,...,x_i,...,x_n),\ T_{Rj},\ D_j]; \quad (i = 1,2,...,n;\ j = 1,2,...,m) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&y_m = \mathcal{R}_m[f_m(x_1,x_2,...x_i,...,,x_n),\ T_{Rm},\ D_m] \qquad (3)
\end{aligned}
$$

ahol T_{Rj} a rendszer adott kimeneti eseményéhez tartozó válaszidejét jelenti, D_j pedig a hozzárendelt relatív határidőt, amely nem szabad túllépje a rendszer specifikációjában meghatározott T_D értéket [6].

$$\forall D_j \leq T_D \qquad (4)$$

A válaszidők eseményenként változhatnak, mert függnek a rendszerbemenetekhez hozzárendelt f_j függvények által elindított taszk futási idejétől (τ_j). Ha feltételezzük, hogy egy ciklusidőn belül mindegyik kimeneti esemény megvalósul, akkor ezeknek az összege végső soron meghatározza az aktuális eseményekhez tartozó T_{Cj} ciklusidőt is (4. ábra).

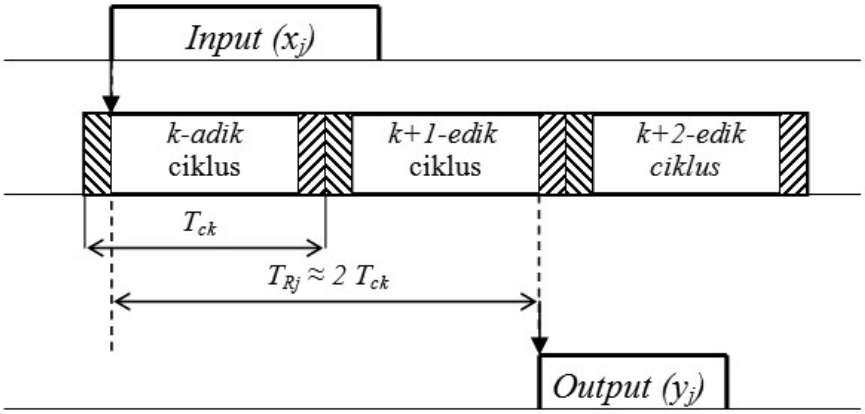


4. ábra. A folyamatok (taszkok) futási ideje

Ha alkalmazzuk az előző fejezetben kapott (2) összefüggést a válaszidőket illetően és feltételezzük, hogy a ciklusidőt jó megközelítéssel *m* számú taszk végrehajtási ideje (τ_j) határozza meg, akkor felírhatjuk a következő összefüggést:

$$T_{Rj} = q \cdot T_{Cj} = q \cdot \sum_{j=1}^{m} \tau_j \le T_{C\,max}; \text{és} \forall\, \tau_j \le D_j \tag{5}$$

A (5) összefüggésben $1 \le q \le 2$, valós szám, nevezzük *bemenet-gerjesztési tényezőnek*, a véletlen időpillanatban érkező bemeneti gerjesztés és a ciklusidőn belüli bementi változók aktualizálási időpontjának viszonyát mutatja [6]. Ha a ciklust megelőzően a rendszerhez nem érkezik egyetlenegy kérés sem vagy egyetlen egy bemenete sincs, akkor $q = 1$, mert a ciklus alatt lefutó taszkok várhatóan még az adott ciklusban eredményt produkálnak. A legkedvezőtlenebb esetben (5. ábra.), amikor a bemeneti gerjesztés éppen lekési az adott ciklus bemeneti aktualizálási idejét, akkor az erre adott válasz csak a következő ciklusban jön létre, vagyis $q = 2$.



5. ábra. A válaszidő alakulása a legkedvezőtlenebb helyzetben

Az előbbi megállapításokat és a (4) és (5) összefüggéseket figyelembe véve kijelenthetjük, hogy *a ciklikus működésű alaprendszer valósidejű, ha a legkedvezőtlenebb eset ciklusideje (a maximális ciklusidő) a határidő felénél nem nagyobb:*

$$T_{C\,max} \le \frac{T_D}{q} = \frac{T_D}{2}; \tag{6}$$

ahol:

$$T_{Cmax} = max(T_{C1}, T_{C2}, \ldots, T_{Ck}, \ldots), \tag{7}$$

T_D pedig a rendszer specifikációjában megadott időkorlát (határidő).
<u>*Következmény:*</u> A (5) és (7) összefüggésekből egyértelműen következik, hogy:

$$\forall T_{Ck} \le \frac{T_D}{2}; \quad (j = 1, 2, \ldots, m), \tag{8}$$

illetve:

$$\forall\, T_{Rj} \le T_D \tag{9}$$

vagyis, *az alaprendszer valósidejűnek tekinthető, ha bármely bemeneti gerjesztésre adott válaszidő sohasem haladja meg az előírt határidőt.*

2.2. A legkedvezőtlenebb eset ciklusidejének meghatározása

Az (5) összefüggés szerint a ciklusidő a taszkok számától és azok időtartamától függ. Szigorúan valósidejű rendszereknél a taszkok időtartamához is hozzárendelhetünk relatív időkorlátokat (D_j). Ebben az esetben viszont ahhoz, hogy a rendszer megtartsa szigorúan valósidejű jellegét, az egyes taszkok futási ideje sem haladhatja meg a hozzájuk tartozó időkorlátot:

$$\forall\, \tau_j \le D_j\,; \quad (j = 1,2,\ldots,m) \tag{10}$$

A taszkok sorrendjét, a legtöbb esetben, az őket kiváltó bemeneti események érkezési sorrendje határozza meg. Egy esemény bekövetkezésének elmaradása esetén, a taszk futása az adott ciklusban akár el is maradhat. Előfordulhatnak még periodikusan futó határidős taszkok vagy kitüntetett taszkok (pl. hibaesemények), amelyeket célszerű prioritással ellátni, tekintettel arra az esetre, amikor azonos időben több kérés is érkezik a rendszer bemeneteire. A prioritásspecifikáció nélküli, azonos időpillanatban kiváltott taszkokat a rendszer véletlenszerű sorrendben hajtja végre. Ezeket a taszkokat bármikor megszakíthatják magasabb prioritással rendelkezőek.

Az előbb elmondottak alapján meghatározható a rendszer maximális ciklusideje, ha feltételezzük, hogy az adott ciklusban mindegyik taszk lefut és mindegyikhez a legkedvezőtlenebb eset időtartamát (C_j) soroljuk. Ha ehhez az esethez rendeljük hozzá a taszkok határidejét (D_j), felírhatjuk a következő összefüggést:

$$T_{C\,max} = \sum_{j=1}^{m} C_j \le \sum_{j=1}^{m} D_j \tag{11}$$

Figyelembe véve az (6) szerinti, valós idejű alaprendszerre vonatkozó megállapítást, vagyis azt, hogy a legkedvezőtlenebb gerjesztési esethez még maximális ciklusidő is társul, akkor:

$$T_{C\,max} \le \frac{T_D}{2} \tag{12}$$

A (10) és (12) összefüggések alapján teljes bizonyossággal kijelenthetjük, hogy *a ciklikusan működő alaprendszer valósidejű, ha a határidős taszkok futási ideje a határidőn belül lezajlik, a maximális ciklusidő pedig a rendszer specifikációjában megadott időkorlát felénél nem nagyobb.*

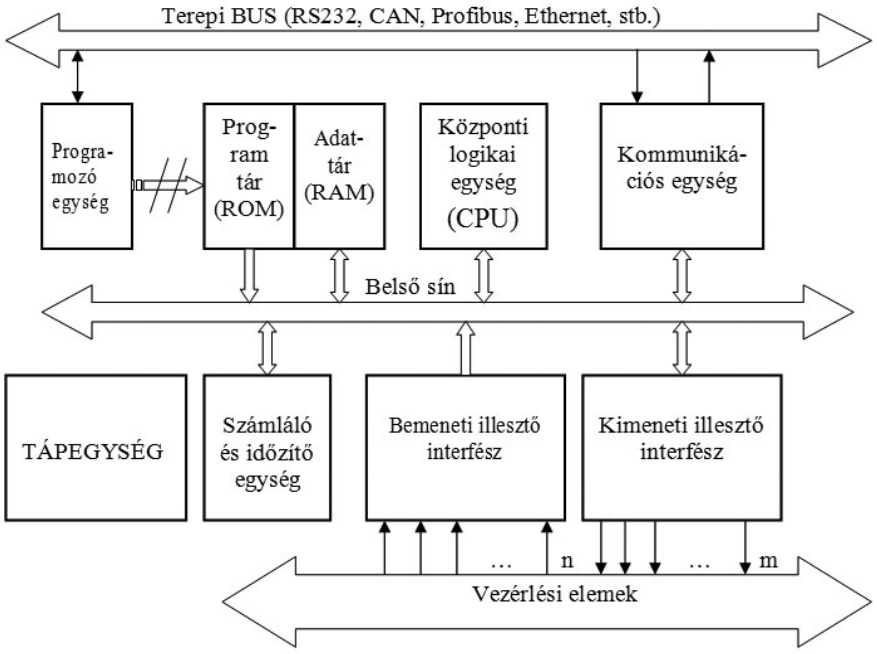
2.3. A programozható logikai vezérlő (PLC) mint valós idejű alaprendszer

A programozható logikai vezérlők gondolata akkor merült fel, amikor a hatvanas években a piacvezető nagy amerikai autógyárak, azzal szembesültek, hogy már nem csak az európai, hanem az amerikai piacon is kezdenek megjelenni az egyre inkább vevői igényeket kielégítő, kisebb teljesítményű, de sokkal hatékonyabban működő japán gépkocsik. Huzalozott vezérlők használatával a technológia rugalmassá tétele meglehetősen nehéz feladat. A gyártó- és szerelősorok átállítása valamint a vezérlő rendszerek módosítása gyakran több napot is igénybe vett. Ezért az akkori piacvezető amerikai autógyártó, a General Motors 1968-ban pályázatot írt ki olyan megoldásra, amely képes kiváltani a relés vezérléseket, könnyen programozható és megbízhatóan működik. A

pályázatot a Modicon cég nyerte el, mérnökei pedig nyomban hozzá is kezdtek a munkához.

Az első PLC Richard Morleyhoz és csapatához fűződik, akik „Modicon 084” néven 1969-ben készítették el. Morley túlzásnak tartotta, hogy computernek hívják, így inkább a controller elnevezést javasolta. Mivel elsősorban logikai műveletek végzésére szánták, és ami a lényeg, hogy programozható volt, ezért a máig is ismert elnevezést tulajdonították neki. Központi egységét huzalozott CPU alkotta, mindösszesen 1 kB memóriával rendelkezett, viszont 128 bemeneti/kimeneti csatornát tudott kezelni [7].

Az elmúlt több mint 40 év alatt a PLC-k sokat változtak. Központi logikai egységüket rendszerint RISC típusú mikroprocesszor (akár több processzor) alkotja azért, hogy a valós idejű működés minél inkább támogatott legyen. A kezelhető memóriaterület is jóval nagyobb, bár nem ez a meghatározó, ha minősíteni kell egy PLC-t. A bemeneti és kimeneti csatornák száma pedig moduláris felépítésüknek köszönhetően, akár több ezer is lehet. A korszerű, moduláris felépítésű PLC-k, a hagyományos kommunikációs kapcsolatokon (RS232, USB, CAN, Profibus, stb.) kívül, Ethernet interfésszel is rendelkeznek, vagy Ethernet modul illeszthető hozzá. (6. ábra)

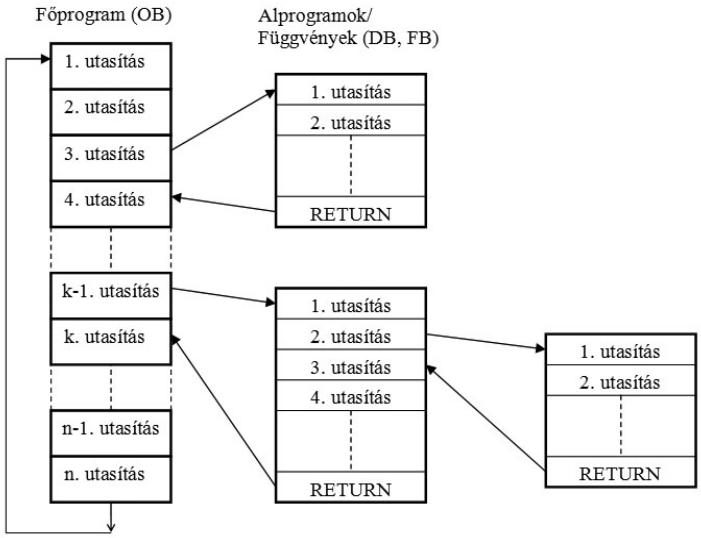


6. ábra. Egy PLC általános felépítése

Azok a programozható logikai vezérlők, amelyek rendelkeznek Ethernet csatlakozási lehetőséggel, felépítésüket illetően belátható, hogy eleget tesznek a valós idejű alaprendszer szerkezetének, mert vannak bemeneti/kimeneti csatornái, amelyeken keresztül kapcsolatot tart a vezérléssel. Most lássuk, hogy programvégrehajtás szempontjából teljesítik-e a (10) és (12) kritériumokat?

A PLC rendszerprogramja ciklikus működést biztosít a felhasználói program számára. Indítás vagy RESET után a rendszerprogram beolvassa a bemeneti változókat és elindítja a főprogram utasításainak szekvenciális végrehajtását. A legtöbb felhasználói program strukturált szervezésű, vagyis a főprogramból szubrutinok (taszkok) vagy függvények hívhatóak. Ezek hívása sok esetben eseményvezérelten történik, de lehetnek periodikusan ismétlődő idővezérelt, vagy prioritással ellátott taszkok is. Az utolsó utasítás után a rendszerprogram az eredményeknek megfelelően aktualizálja a kimeneti csatornákat és visszatér a főprogram elejére (7. ábra). Ezt az egyszeri program-végrehajtási időt

nevezzük a PLC ciklusidejének vagy letapogatási időnek (scan time), amely mindenben azonosnak tekinthető a 1.3. alfejezetben meghatározottal.



7. ábra. A felhasználói program szerkezete

Látható, hogy a ciklusidő az utasítások számától, az utasítások bonyolultságától és természetesen a processzor művelet-végrehajtási sebességétől függ. Nem túlságosan bonyolult vezérlési programoknál ez néhány milliszekundumtól, 10-50, legfeljebb 100 ms-ig terjed.

A bementi gerjesztésre adott válaszidők jó esetben egy, de maximum két ciklusidőt igényelhetnek. Ennek megfelelően a határidőt minden esetben tartani tudják, ha a ciklusidőt sikerül a határidő felénél nem nagyobb értéken tartani. Egyes PLC-k ezt képesek ellenőrizni is és „hibaüzenettel" reagálnak ennek túllépésekor. Egy ilyen hibaüzenet természetesen nem a PLC leállását jelenti. Ilyenkor meghív egy olyan taszkot, amely tartalmazza, hogy ilyen esetben mit kell, hogy tegyen a PLC. Ennek ismeretében a továbbiakban a ciklikus működésű, Ethernet hálózati kapcsolattal rendelkező PLC-t valós idejű alaprendszernek tekintem [8].

3. Nem ciklikus működésű valós idejű rendszerek

Az előző fejezetben meghatározott ciklikus működésű, valós idejű alaprendszer legfontosabb tényezője a ciklusidő. A taszkok sorrendje és azok határideje legfeljebb az optimális ciklusidő elérése érdekében lehet érdekes, mert az általuk kiváltott hatás amúgy is a ciklus végén kerül érvényesítésre. A határidő túllépés – megfelelő intézkedéseket követően – rendszerint a taszk megszakítását eredményezi azért, hogy a többi zavartalanul lefuthasson. A taszkok futási idejét, így végső soron a ciklusidőt alapvetően a rendszert alkotó CPU (vagy CPU-k) műveletvégzési sebessége határozza meg. Ezért nem mindegy, hogy a taszkok milyen sorrendben kerülnek feldolgozásra. A cél, hogy a prioritási sorrend betartása mellett, a processzor kihasználtság a legjobb legyen.

A valós idejű rendszerek nem csak ciklikus működésűek lehetnek. Az ilyen rendszerek valós idejű működése sokkal összetettebb feltételek teljesítését kell biztosítsák. Ezek a rendszerek elsősorban arra törekednek, hogy taszkjaik optimálisan ütemezhetőek legyenek adott időintervallumon belül. Több megközelítés is született ebben a témában, főleg egyprocesszoros rendszerekhez. Ezek közül megemlíteném az *RM (Rate Monolitic) ütemezést* és ennek egy továbbfejlesztett formáját az EDF (*Earliest Deadline First)*

algoritmust. Az RM algoritmus elvének kidolgozása Liu és Layland nevéhez fűződik [10]. Az eljárás egyprocesszoros rendszert feltételez, amelyben minden taszk periodikus és a gyakrabban futó taszkok magasabb prioritást élveznek. Az elv a processzor igény és a processzor kihasználtság közötti relációkon alapul [9]. A működés preemptív, vagyis a magasabb prioritással rendelkező taszk megszakíthatja az alacsonyabb prioritásút.

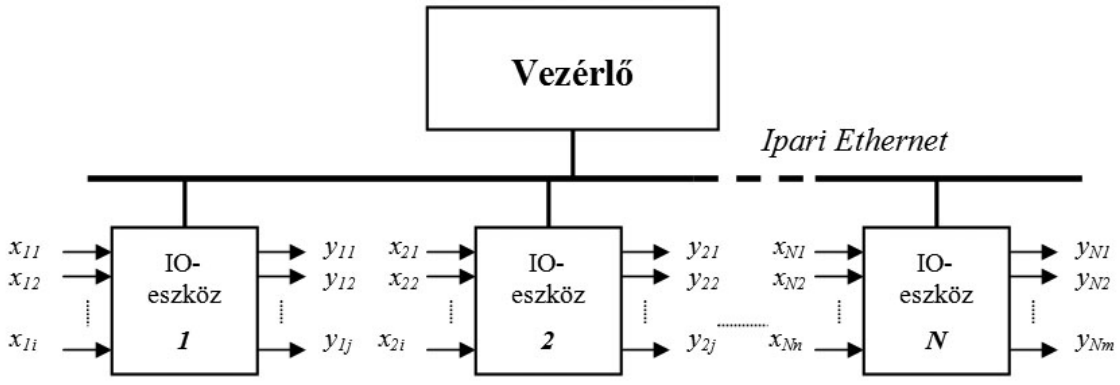
A módszernek egyik hátránya, hogy nem garantálható a teljes processzor-kihasználtság. Másrészt sztatikus jellegű, mivel a taszkok prioritása a periódusuk szerint van meghatározva, és a prioritást illetően egyáltalán nem mondható rugalmasnak. Ráadásul az optimális ütemezés csak akkor biztosítható, ha a taszkok határideje megegyezik a taszkok periódusával.

Az EDF ütemezési algoritmus az előbbinél egy sokkal rugalmasabb, dinamikus megoldást eredményez. A prioritás a taszkok abszolút határidejéhez van kötve, vagyis azé a magasabb prioritás, amely a legkorábban lejáró határidővel rendelkezik. Innen ered az elnevezés is. Dinamikusságának köszönhetően, amennyiben új, végrehajtásra kész folyamat jelentkezik, a prioritási sorrend újragenerálódik és lehetővé teszi, hogy a magasabb prioritású taszkok akár meg is szakíthatják az alacsonyabb prioritásúak futását [11].

4. Valós idejű ipari Ethernet alrendszer

Ha a 2. fejezetben meghatározott alaprendszert kiterjesztjük az ipari Ethernet hálózatokra, akkor meghatározhatjuk azokat a körülményeket és feltételeket, amelyek mellett, az egyébként nem determinisztikus Ethernet hálózat alkalmassá válik valósidejű feladatokat ellátó egységek közötti kommunikáció lebonyolítására. Ennek megfelelően a valós idejű Ethernet alrendszert úgy tekintjük, mintha az alaprendszer bemeneti és kimenetei eseményeit kezelő csatornák eszközei Ethernet hálózaton keresztül kapcsolódnak a folyamatokat (taszkokat) irányító vezérlőhöz. Feltételezzük, hogy a vezérlő és a perifériák közötti viszonyt valamilyen időhöz kötött, token passing vagy master-slave vagy provider-consumer típusú megoldás biztosítja. Ez utóbbi, olyan szempontból előnyösebb, hogy duplex hálózatban, bármelyik fél kezdeményezhet kommunikációt egymástól függetlenül.

Feltételezzük, hogy az n számú bemeneti (x_i) és az m számú kimeneti (y_j) csatorna N számú Ethernetes periféria eszközön (IO-eszköz) keresztül kapcsolódik a vezérlőhöz. A következő ábrán egy ilyen Ethernet alrendszert láthatunk (8. ábra).



8. ábra. Valós idejű Ethernet alrendszer

Az előző fejezetben láthattuk, hogy ahhoz, hogy kiszámítható működésűvé tegyük az egyébként nem determinisztikus Ethernet hálózatot, egyik alapvető feltétel az, hogy

valamilyen jól meghatározott időközönként épüljön ki kapcsolat az eszközök és a vezérlő között. Egymástól függetlenek legyenek, rendelkezzenek prioritással a nem valósidejű (NRT) kommunikáció felett, de számottevően ne akadályozzák azokat.

4.1 A buszfrissítési ciklus

Egyik módja a ciklikusan működő kapcsolati rendszereknek, a nem Ethernet alapú terepi buszrendszereknél jól bevált master-slave megoldás, amikor a master a konfigurációjában meghatározott időközönként lekérdezi a slave eszközök bemeneteit, illetve adatokat küld ezeknek a kimeneteire. Egy cikluson belül akár az összes eszköz lekérdezésre kerülhet adott sorrendben, jól meghatározott időrések szerint. Ha a sorrend nem változik, akkor gyakorlatilag mindegyik eszköznek azonos ciklusidő szerint van lehetősége adatot cserélni a masterrel. A módszer igazából akkor alkalmazható a leghatékonyabban, amikor az eszközök hasonló paraméterekkel rendelkező céleszközök működtetését végzik [12].

A másik megoldás, amikor az eszközök és a vezérlő között egymástól független kommunikációs kapcsolatok épülnek fel (provider-consumer), vagyis mindegyik eszközhöz hozzárendelhető saját, egyéni buszfrissítési ciklusideje [13]. Ez főleg akkor előnyös, ha az eszközökhöz kapcsolódó rendszerek más-más időspecifikációt igényelnek. Egy lassú folyamatot felesleges olyan ütemben lekérdezni, mint egy nagyon gyors folyamatot. Ezáltal lehetőség nyílik az adott sávszélesség jobb kihasználására, ami lehetővé teszi, hogy nagyszámú IO-eszköz esetén is a valósidejű elvárásoknak megfelelően működjön a rendszer. Természetesen ez a megoldás is lehetőséget biztosít arra, hogy azonos funkcionalitású céleszközök kiszolgálását biztosítsa nagy precizitással. Ennek érdekében, az egyébként sajátságos egyéni ciklusidőket azonossá kell tegyük és szinkronizálni kell, hogy minél precízebb beavatkozásokat biztosíthassunk a célrendszerek számára, ott ahol ezek megkövetelik.

Bármelyik megoldást is feltételezzük a busz ciklus meghatározó tényezője lesz a valósidejű Ethernet alrendszernek. A szigorúan valósidejű rendszereknél ez a ciklusidő megfelelően rövid és pontos kell, hogy legyen. Sok esetben nem is az a probléma, hogy viszonylag nagy a kimenetek reakcióideje, hanem az, hogy ezek minél pontosabbak legyenek, mert akkor mindig ugyanazzal a késéssel tudunk kalkulálni.

Felhasználva a (3) összefüggését és figyelembe véve az egyes eszközök busz ciklusidejét, a 8. ábra szerinti rendszer esetében bármelyik kimenetre felírhatjuk:

$$y_{kj} = \mathcal{R}_{kj}[f_{kj}(x_{11},\ldots,x_{ki},\ldots,x_{Nn});\mathcal{T}_k,\ T_C,\ T_{Dk}];\ \ (i = 1,2,\ldots,n;\ j = 1,2,\ldots,m;\ k = 1,2,\ldots,N); \quad (13)$$

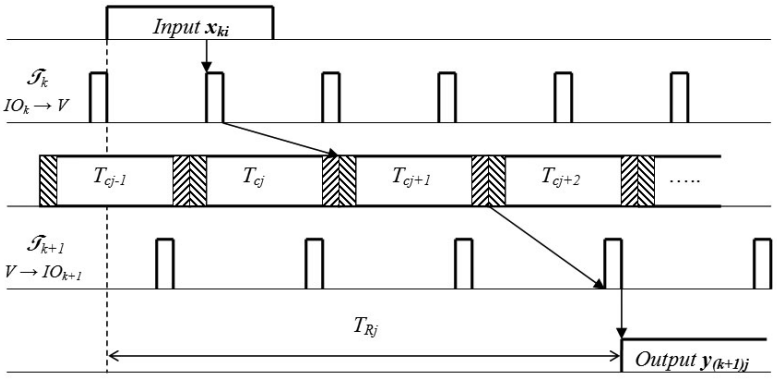
ahol $\mathcal{T}_k$ a k-adik eszköz buszfrissítési ciklusideje, T_C a vezérlő ciklusideje, T_{Dk} pedig a k-adik eseményhez tartozó relatív határidő.

Tételezzük fel, hogy az $y_{(k+1)j}$ eseményt az x_{ki} gerjesztés váltja ki, az eszközökhöz hozzárendelt buszfrissítési ciklusidők pedig különbözőek ($\mathcal{T}_k \neq \mathcal{T}_{k+1}$). A legkedvezőtlenebb eset (9. ábra) reakcióidejének meghatározására a következő relációt írhatjuk fel:

$$T_{Rj} = 2 \cdot T_C + \mathcal{T}_k + \mathcal{T}_{k+1}; \quad (14)$$

A (14) reláció egy általános helyzet reakcióidejét határozza meg, amikor a kommunikációs relációban résztvevő eszközök buszfrissítési ideje eltér egymástól, a vezérlő program-végrehajtási ciklusideje pedig tetszőleges. Láthatjuk, hogy a vezérlő ciklusideje súlyozottan befolyásolja a válaszidőket. Ezért nem is igazán érdemes a buszfrissítési ciklusokat sokkal kisebbre választani, mint a vezérlő ciklusideje, mert, ha

lassúbb a feldolgozás, a gerjesztések felülíródhatnak, a kimenetek pedig nem frissülnek a vevő buszciklusa szerint. Mivel a vezérlő ciklusidejét a taszkok futási idejének összege határozza meg, azaz adott, ennek megfelelően: $\mathcal{T}_{kmin} \geq T_C$. Vagyis az alrendszer minimális buszfrissítési idejét a vezérlő ciklusidejéhez kell igazítani (9. ábra).



9. ábra. A legkedvezőtlenebb helyzet reakcióideje

Egy másik, ugyancsak szélsőséges esetnek tekinthető az a szituáció, amikor a buszfrissítési idők jóval nagyobbak, mint a vezérlő ciklusideje. Ilyenkor egyértelmű, hogy a válaszidőket csak a buszfrissítési idők nagysága határozza meg, vagyis függetlenek a vezérlő ciklusidejétől.

4.2. A valós idejű Ethernet alrendszer meghatározása

Ahhoz, hogy az ipari Ethernet alrendszert valósidejűnek tekinthessük, teljesítenie kell a valósidejű alaprendszer (12) szerinti feltételeit, vagyis:

$$T_{Cmax} \leq T_{Dk}/q \tag{15}$$

Ahol a (11) alapján feltételezzük, hogy:

$$T_{Cmax} = max(T_C, \mathcal{T}_k, \mathcal{T}_{k+1}) = \mathcal{T}_k \tag{16}$$

A (14) összefüggést alkalmazva a legkedvezőtlenebb esetre, amikor szinkronizálás nélküli a kommunikáció, továbbá a gerjesztés olyan időpillanatban történik, hogy mindkét irányban éppen lekési a buszfrissítési időt és a bemeneti aktualizálást is egyaránt, következik, hogy ebben az esetben a gerjesztésre adott válasz reakcióideje kb. négyszerese lesz a legnagyobb busz ciklusidőnek. (9. ábra). Ebben az esetben tehát: $q = 4$. Következik, hogy:

$$\forall\ T_{Rj} = q\ T_{Cmax} = 4\ \mathcal{T}_k \leq T_{Dk} \tag{17}$$

Kijelenthetjük, hogy *az Ethernet alrendszer valósidejű, hogyha a legnagyobb ciklusidővel rendelkező eszköz buszfrissítési periódusa legalább négyszer kisebb, mint a rendszer specifikációjában hozzárendelt időkorlát.*

Megjegyzés: Mivel $q = 4$ helyzet bekövetkezési valószínűsége igen csekély, a gyakorlatban, a legtöbb esetben elegendő ha $q = 3$, de természetesen lehetőség van ettől eltérő értékek kiválasztására is. A valóságban ez azt jelenti, hogy ha három buszfrissítési idő elteltével sem érkezik válasz a kérésre, akkor gyanítható, hogy valamilyen

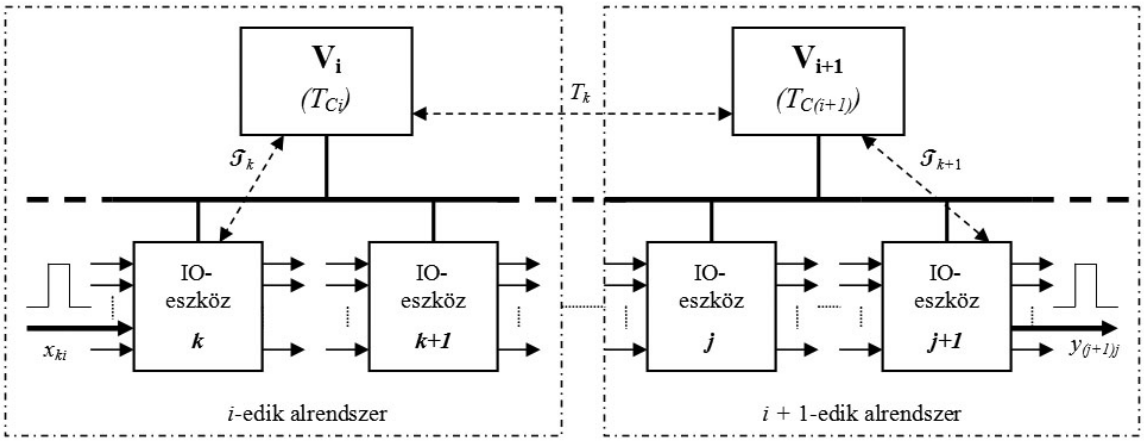
hibaesemény történt, és a rendszer ennek megfelelően kell reagáljon. Az *n, m, N* mennyeiségekre vonatkozó korlátok tágabb értelmezés szerint a vezérlő kapacitásától függnek, de szigorúbb valósidejű feltételek mellett természetesen a rendszerspecifikációban szereplő T_D időkorlát is meghatározó lehet, főleg az eszközök számát illetően.

A (17) alapján elmondható, hogy a feltételezett ipari Ethernet alrendszer valósidejű egységes egészként működő valósidejű alaprendszernek tekinthető annak ellenére, hogy a bemeneti és kimeneti változók más és más, egymással Ethernetes kapcsolatban álló IO eszközhöz tartoznak. A valósidejű feltételek teljesülését egyértelműen a buszfrissítési ciklus helyes megválasztása és megfelelő értéken tartása határozza meg. Ahogy szigorítjuk az alrendszer valósidejű követelményeit, úgy egyre szigorúbb feltételeket kell szabnunk a ciklusidőket illetően is. Szigorúan valósidejű rendszereknél arra kell törekednünk, hogy a működés minél kiszámíthatóbb legyen, azaz a 9. ábra szerinti legkedvezőtlenebb helyzeteket elkerüljük. Ezt úgy tudjuk elérni, hogy szinkronizáljuk az eszközök ciklusidejét. Ezáltal egyrészt az eszközök ciklusidő periódusát egyenlővé tesszük, másrészt igyekszünk a jittert (ciklusidő eltérés) minél kisebbé tenni [14].

5. A valósidejű Ethernet alrendszer kiterjesztése

Ha kettő vagy több, egymástól független funkcionalitású valósidejű Ethernet alrendszert a hálózaton keresztül összekapcsolunk egymással, egységes egészként működő valósidejű Ethernet rendszert kapunk, amely teljesíteni fogja a (9), (10) összefüggésekben meghatározott követelményeket a megfelelő feltételek mellett.

A következő ábrán (10. ábra) két tetszőlegesen választott alrendszer kapcsolatát látjuk. Feltételezzük, hogy az *i*-edik alrendszer kommunikációs relációt épít a *i+1*-edik alrendszerrel. Továbbá a *k*-adik IO eszköz valamelyik bemenete gerjeszti azt az eseményt, amelyik a *j+1*-edik eszköz kimenetén kell megvalósuljon.



10. ábra. Két tetszőleges alrendszer kommunikációs kapcsolata.

Legyen T_{Ci} illetve $T_{C(i+1)}$ a két vezérlő ciklusideje, ϑ_k valamint ϑ_{k+1} az érintett IO eszközök és T_k a két vezérlő közötti buszfrissítési idők. A (14) relációt kiterjesztve az így létrejövő rendszerre, következik:

$$T_{Rj} = 2\cdot(T_C + T_{C(i+1)}) + T_k + \vartheta_k + \vartheta_{k+1}; \qquad (18)$$

Feltételezzük, hogy a (16) relációnak ugyancsak $\mathcal{T}_k$ tesz eleget, tulajdonképpen a (17) összefüggést kapjuk $q = 7$-re.

$$\forall\ T_{Rj} = q\ T_{Cmax} = 7\ \mathcal{T}_k \leq T_{Dk} \tag{19}$$

Kijelenthető, hogy *az Ethernet hálózaton keresztül egymással összekapcsolt ciklikusan működő alrendszerek valós idejű Ethernet rendszert alkotnak, ha a legnagyobb ciklusidővel rendelkező eszköz buszfrissítési periódusa legalább hétszer kisebb, mint a rendszer specifikációjában meghatározott időkorlát.*

IRODALOMJEGYZÉK

[1] Jane W. S. Liu: Real-Time Systems, Prentice Hall, 2000

[2] Tei-Wei Kuo: Introduction to Real-time Process scheduling, National Taiwan University http://www.csie.ntu.edu.tw/~ktw/rts/ch-short-course-uni.pdf

[3] C. L. Liu, J. W. Layland: Scheduling algorithm for multiprograming in a Hard Real-time Environments, Journal of ACM, 1973.

[4] Kondorosi Károly, Szirmay-Kalos László, László Zoltán: *Objektum orientált szoftverfejlesztés,* 5. fejezet, http://www.tankonyvtar.hu/hu/tartalom/tkt/objektum-orientalt/ch05.html#id517610

[5] Ajtonyi István: Ipari kommunikációs rendszerek III., Aut-Info Kft. Miskolc, 2009.

[6] Ferenczi István: *Profinet IO kontroller ciklusidő változásának vizsgálata,* XXV. MicroCAD Nemzetközi Konferencia kiadvány, 2010. március.

[7] Schneider magazin, VIII. évfolyam, 1. szám, 2008. február.

[8] Ferenczi István: *Programmable Logic Controllers as Real-Time Industrial Ethernet Base System Devices,* International Symposium on Applied Informatics and Related Areas: AIS 2013, Óbudai Egyetem Székesfehérvár, Magyarország, 2013.11.07-2013.11.09. pp. 46-49. ISBN 978-615-5018-88-6.

[9] Tei-Wei Kuo: Introduction to Real-time Process scheduling, National Taiwan University http://www.csie.ntu.edu.tw/~ktw/rts/ch-short-course-uni.pdf

[10] C. L. Liu, J. W. Layland: Scheduling algorithm for multiprograming in a Hard Real-time Environments, Journal of ACM, 1973.

[11] S. K. Baruah, A. K. Mok, L. E. Rosier: Preemtively scheduling Hard Real-Time sporadic task on one processor, IEEE Real-Time System Symposium, 1990.

[12] Ajtonyi István, PLC és SCADA-HMI rendszerek II. & Ipari kommunikációs rendszerek II. Aut-Info Kft. ISSN 1789-5456, ISBN 978-963-661-833-9, pp. 408, 2008. május.

[13] Ferenczi István: *Válaszidők vizsgálata egy Profinet IO rendszernél*, Proceedings of Factory Automation 2010: 15-16th April, 2010, Kecskemét, Hungary. pp. 105-110. ISBN 978-963-7294-83-9.

[14] Ferenczi István: *Szinkronizált adatátviteli késleltetések elemzése a Profinet IRT rendszernél,* XI. ENELKO – XX SzámOkt Nemzetközi Energetikai-elektrotechnikai és Számítástechnikai Konferencia, Szatmárnémeti, Románia, 2010.október 7-10, pp. 118-123. ISSN 1842-4546.