Digitális technika 2.

BMEVIIIAA06

10. előadás

Kommunikáció, pufferkezelés, DMA

Láttuk eddig:

- A kommunikációs perifériák megoldják adatok soros küldését/fogadását.
- Megtanultunk két jellemző soros kommunikációs perifériát:
  - SPI
  - UART
- A kommunikációs perifériáknak általában valamilyen átmeneti tárolójuk (pufferük) is van.

Nagy mennyiségű adat küldése/fogadása

- Írjunk egy programot, amely az UART1-en folyamatosan (amilyen gyorsan csak lehet) elküld egy 1000 bájt hosszú, a memóriában az „tomb” címkétől kezdődő tömböt.
- Az UART1 bitsebessége legyen 19200 bps
- 8 adatbit, nincs paritás, 1 stop bit.
  1. inicializáljuk az uart-ot (feltehetjük, hogy a perifériát már a megfelelő portlábhoz hozzárendelték a PPS-ben)
  2. írjuk meg a szubrutint, ami a küldést végzi

UART inicializálása

- Az UART bitsebessége legyen 19200 bps
- 8 adatbit, nincs paritás, 1 stop bit.

1. inicializáljuk az UART-ot

```
Uart_init:
        mov ??????,w0
        mov w0,U1MODE     ; üzemmód beállítás
        mov ??????,w0
        mov w0,U1STA      ; üzemmód (és státusz)
        mov ??????,w0
        mov w0,U1BRG      ; Baudrate állítás


        return
```

U1MODE regiszter 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UARTEN	-	USIDL	IREN	RTSMD	-	UEN1	UEN0	WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL1	PDSEL0	STSEL
1		0	0	0		0	0	0	0	0	0	0	0	0	0

UARTEN	indítás/leállítás	1 = UART engedélyezve 0 = UART tiltva
USILD	Működés IDLE módban	1 = Az UART IDLE módban leáll 0 = Az UART IDLE módban is működik
IREN	IRDA vevő mód	1 = IRDA (infravörös távirányító protokoll) mód 0 = „hagyományos” UART mód
RTSMOD	RTS kimenet módja	1 = Adás engedélyezés mód 0 = Hagyományos RTS mód
UEN [1:0]	UART-hoz tartozó portlábak engedélyezése	11: TX, RX és órajel 10: TX, RX, RTS és CTS 01: TX, RX és RTS 00: csak TX és RX

WAKE	„Felébresztő megszakítás”	1 = Az UART RX bemenetén érkező lefutó él azonnal megszakítást okoz (a processzor azonnal felébreszthető kommunikáció kezdetekor) 0 = Csak tényleges adat vétele okoz megszakítást
LPBACK	Teszt mód	1 = A kiküldött adatok visszaérkeznek, mintha vett adatok lennének 0 = Normális működés
ABAUD	Baud rate automatikus megállapítása	1 = A következő karakter vételéből megméri a baud rate-et (csak 0x55 karakterrel működik). A mérés után a bit 0-ba vált.
URXINV	Vétel láb polaritás megfordítása	1 = RX nyugalomban ’0’ 0 = RX nyugalomban ’1’ (hagyományos UART)
PDSEL	Adathossz és paritás állítás	11 = 9 bites adat, nincs paritás 10 = 8 bites adat, páratlan paritás 01 = 8 bites adat, páros paritás 00 = 8 bites adat, nincs paritás
STSEL	Stop bit hossza	1 = 2 db stop bit 0 = 1 db stop bit

BRGH=0

$$\text{Baud Rate} = \frac{F_{CY}}{16 \cdot (UxBRG + 1)}$$

BRGH=1

$$\text{Baud Rate} = \frac{F_{CY}}{4 \cdot (UxBRG + 1)}$$

U1STA regiszter 0x0400

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R-0	R-1	R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
UTXISEL1	UTXINV	UTXISEL0	URXEN	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1,0		ADDEN	RIDLE	PERR	FERR	OERR	URXDA

UTXISEL [0:1]	Adási megszakítás kérésének módja	11 = Érvénytelen Megszakítást kér, ha: 10 = Egy beírt adat küldése elkezdődött 01 = Minden küldés befejeződött 00 = Van hely a küldő pufferben (beírhatjuk a következő adatot)	**URXISEL [0:1]**	Vételi megszakítás kérésének módja	Megszakítást kér, ha: 11 = A vételi puffer tele van (további vétel adatvesztést okozna) 10 = A vételi bufferben már csak 1 adatnak van hely 0- = Van ki nem olvasott adat a vételi pufferben
UTXINV	Adás láb polaritás megfordítása	1 = TX nyugalomban ’0’ 0 = TX nyugalomban ’1’ (hagyományos UART)	**ADDEN**	9. bit címként értelmezése	1 = 9 bites módban a 9. bit 1-es értéke esetén az adatot címként értelmezi 0 = nem kezeli speciálisan a 9. bitet
URXEN	Vétel engedélyezés	1 = Vétel engedélyezve 0 = Vétel tiltva	**RIDLE**	Nincs vétel folyamatban	1 = Nincs vétel folyamatban 0 = Éppen vétel van folyamatban
UTXBRK	BREAK küldőés	1 = BREAK karakter küldése, automatikusan 0-ba áll ha befejeződött	**PERR**	Paritáshiba	1 = A vételi pufferben aktuálisan olvasható adat paritáshibás volt 0 = nem volt ilyen hiba
UTXEN	Adás engedélyezés	1 = Adás engedélyezve 0 = Adás tiltva	**FERR**	Kerethiba	1 = A vételi bufferben aktuálisan olvasható adat keretezés hibás volt 0 = nem volt ilyen hiba
UTXBF	Adó buffer tele jelzés	1 = Az adó buffer tele 0 = Van hely az adó bufferben	**OERR**	Túlfutás	1 = Adatvesztés történt, mert adat érkezett, de a vételi pufferben nem volt hely 0 = nem volt ilyen hiba
TRMT	Adás vége jelzés	1 = Nincs adás folyamatban, minden korábbi küldés befejeződött 0 = Adás folyamatban	**URXDA**	Vett adat elérhető	1 = A vételi pufferben van olvasható adat 0 = A vételi puffer üres

U1BRG regiszter

- Fcy=16MHz
- Baud rate = 19200

$$\text{Baud Rate} = \frac{\text{FCY}}{16 \bullet (\text{UxBRG} + 1)}$$

- $\text{U1BRG}=\frac{16000000}{16 \cdot 19200} - 1 = \frac{10000}{192} - 1 = 51{,}083$

- U1BRG=51
- Baj-e, ha ez nem kerek?
  - Az igazi baud rate most így 51-el: 19230,77, ez 0,16% hiba
  - Mivel minden 10 bitenként úgyis újra szinkronizálunk, ez csak azt jelenti, hogy a 10. bit 1,6%-kal „előrébb lesz”, mint kellene.
  - A bitidő közepén vett minta még így is bőven jó lesz.

UART inicializálása

- Az UART bitsebessége legyen 19200 bps
- 8 adatbit, nincs paritás, 1 stop bit.
  1. inicializáljuk az uart-ot

```
Uart_init:
        mov #0x8000,w0
        mov w0,U1MODE
        mov #0x0400,w0
        mov w0,U1STA
        mov #51,w0
        mov w0,U1BRG


        return
```

Uart küldő szubrutin

- Írjunk egy programot, amely az UART1-en folyamatosan (amilyen gyorsan csak lehet) elküld egy 1000 bájt hosszú, a memóriában a „tomb” címkétől kezdődő tömböt.

Lépések:

1. Várjuk meg, hogy van-e hely az UART küldő pufferben
2. Tegyünk be egy adatot
3. Ha kell még küldeni kezdjük elölről

UART küldő szubrutin

```
Uart_send_array:
      mov #tomb,w0     ; A tömb kezdőcímét töltsük w0-ba
                       ; ez lesz a pointer
      mov #1000,w1     ; 1000 adatot akarunk küldeni
ciklus:                ; várjunk, amíg nem küldhetünk
                       ; ha van hely a pufferben, küldhetünk
      btst ????

      return
```

U1STA regiszter

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-1	R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
UTXISEL1	UTXINV	UTXISEL0	URXEN	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1,0		ADDEN	RIDLE	PERR	FERR	OERR	URXDA

UTXISEL [0:1]	Adási megszakítás kérésének módja	11 = Érvénytelen Megszakítást kér, ha: 10 = Egy beírt adat küldése elkezdődött 01 = Minden küldés befejeződött 00 = Van hely a küldő pufferben (beírhatjuk a következő adatot)
UTXINV	Adás láb polaritás megfordítása	1 = TX nyugalomban ’0’ 0 = TX nyugalomban ’1’ (hagyományos UART)
URXEN	Vétel engedélyezés	1 = Vétel engedélyezve 0 = Vétel tiltva
UTXBRK	BREAK küldőés	1 = BREAK karakter küldése, automatikusan 0-ba áll ha befejeződött
UTXEN	Adás engedélyezés	1 = Adás engedélyezve 0 = Adás tiltva
UTXBF	Adó buffer tele jelzés	1 = Az adó puffer tele 0 = Van hely az adó pufferben
TRMT	Adás vége jelzés	1 = Nincs adás folyamatban, minden korábbi küldés befejeződött 0 = Adás folyamatban

URXISEL [0:1]	Vételi megszakítás kérésének módja	Megszakítást kér, ha: 11 = A vételi puffer tele van (további vétel adatvesztést okozna) 10 = A vételi pufferben már csak 1 adatnak van hely 0- = Van ki nem olvasott adat a vételi pufferben
ADDEN	9. bit címként értelmezése	1 = 9 bites módban a 9. bit 1-es értéke esetén az adatot címként értelmezi 0 = nem kezeli speciálisan a 9. bitet
RIDLE	Nincs vétel folyamatban	1 = Nincs vétel folyamatban 0 = Éppen vétel van folyamatban
PERR	Paritáshiba	1 = A vételi pufferben aktuálisan olvasható adat paritáshibás volt 0 = nem volt ilyen hiba
FERR	Kerethiba	1 = A vételi bufferben aktuálisan olvasható adat keretezés hibás volt 0 = nem volt ilyen hiba
OERR	Túlfutás	1 = Adatvesztés történt, mert adat érkezett, de a vételi pufferben nem volt hely 0 = nem volt ilyen hiba
URXDA	Vett adat elérhető	1 = A vételi pufferben van olvasható adat 0 = A vételi puffer üres

UART küldő szubrutin

```
Uart_send_array:
        mov #tomb,w0    ;A tömb kezdőcímét töltsük w0-ba, ez lesz a pointer az olvasáshoz
        mov #1000,w1    ;1000 adatot akarunk küldeni
        mov #U1TXREG,w2 ;ide akarunk írni, de lehet, hogy egyszerűbb indirekt módon
ciklus:
     1  btst U1STA,#UTXBF
     1  bra NZ, ciklus
     1  mov.b [w0++],[w2]  ;bájtosan másoljuk a tömb adott elemét az U1TXREG-re
     1  dec w1,w1
     2  bra nz, ciklus
        return
```

Meddig tart ez a program:

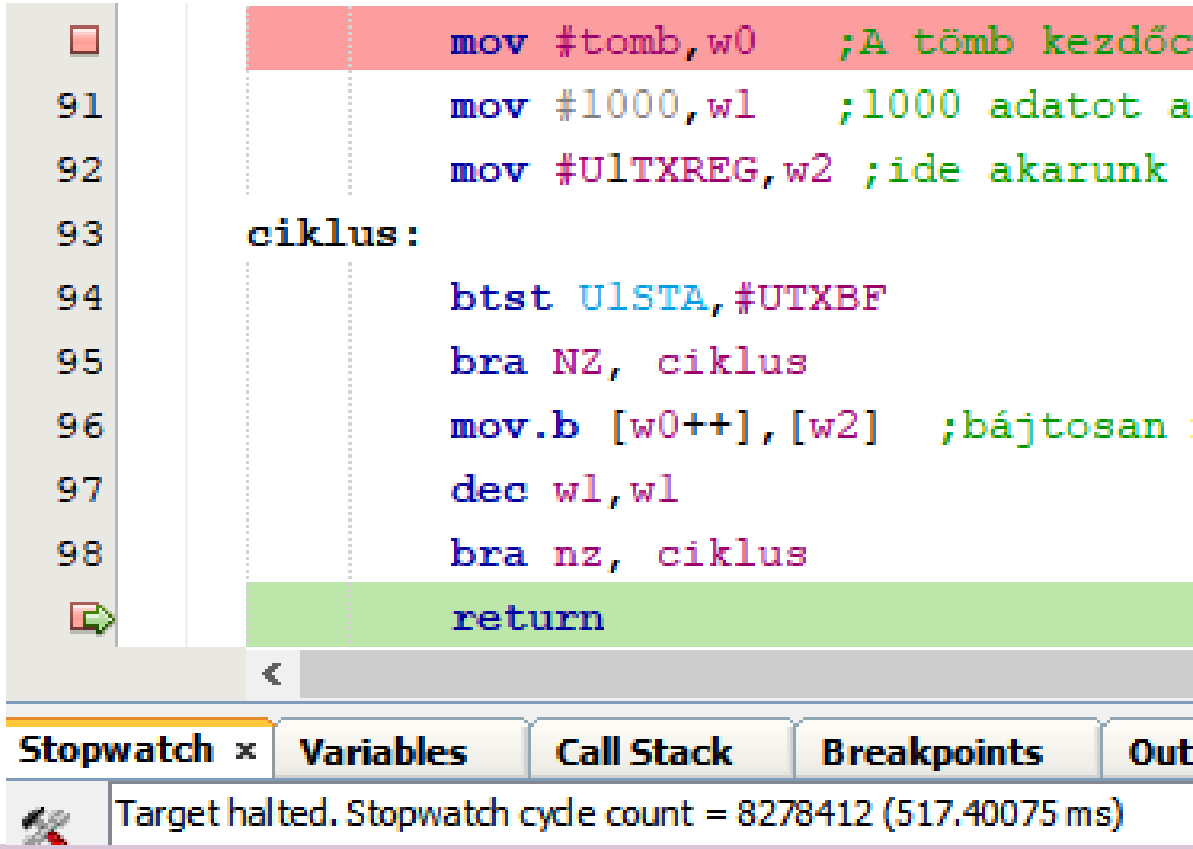
- Ha sose kellene várni az UART-ra, akkor 6 ciklus*1000db +(8)

  →375,5µs

  De sajnos kell várni…

UART küldés várakozással

- Az UART-on az 1000 bájt elküldése 192(30,77) bps-el:
  - Minden egyes bájt +start+stop bit: 10 bitidő
    - →~520µs (már több, mint ami az egész program, várakozás nélkül lenne)
  - 1000 bájt van:
    - →kb: 520ms
  - Az utolsó 4 bájtot nem kell megvárni (ekkora a puffer)
    - →ezért kicsit kisebb



Szoftveres várakozás és kihasználtság

- A teljes adatküldés 520ms
- A processzor ezzel 517,4ms-ig foglalkozik.
- Azaz a processzor „foglaltsága”: 99,5%

Ezalatt igazából:

- 375,5µs-ot „dolgozik” a processzor a többit csak vár, de addig se tud mást csinálni
- A tényleges kihasználtság tehát szörnyű: 0,00007%

UART küldés, megszakítással

- Láttuk, hogy a szoftveres várakozás csak akkor éri meg, ha épp nem lehetne mást csinálni, itt is erről van szó.
- Az UART kérhet megszakítást is.
- Ekkor az „elküldést végző” szubrutinnak 2 része van:
  1. A küldés inicializálása és a megszakítás engedélyezése
  2. A megszakítási szubrutin, ami a tényleges küldést végzi

  A két rész globális változókon keresztül kommunikál:

```
.bss
        kuldoptr: .space 2  ;ez majd az első még el nem küldöttre mutat
        meg_kuldendo: .space 2 ;ez a még hátralévő adatok számát tartalmazza
```

UART küldés megszakítással

- A küldés függvény most csak beállítja a globális változókat és engedélyezi a megszakítást:

```
Uart_send_array_INT:
        mov #tomb,w0
        mov w0,kuldoptr ;kezdetben az aktuális küldendő a tömb eleje
        mov #1000,w0    ;1000 adatot akarunk küldeni
        mov w0,meg_kuldendo

        bset IEC0,#U1TXIE    ; engedjük a megszakítást
                             ; ne töröljük előtte le a flag-et mert az pont jól áll
                             ; a puffer üres, ezért IT-t szeretne kérni
        return
```

UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]

UART küldés megszakítással

- A megszakítás végzi a tényleges küldést:

```
__U1TXInterrupt:
        bclr IFS0,#U1TXIF ; most töröljünk flag-et
        push w0
        mov.b kuldoptr,WREG ;küldjünk 1 bájtot
        mov.b WREG,U1TXREG

        inc kuldoptr     ;pointert növeljük

        dec meg_kuldendo ;egyet elküldtünk

        bra nz,kellmeg_txIT
        bclr IEC0,#U1TXIE ;ha nem kell többet küldeni, akkor tiltsuk le az IT-t
kellmeg_txIT:
        pop w0
        retfie
```

UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]

UART küldés megszakítással

- Ez így meddig tart?
- Az előkészítés (`Uart_send_array_INT`) csak egyszer fut:
  - 10 ciklus → 625ns
- A megszakítás 1000-szer fut le
  - Minden lefutáskor 3 utasításciklus ideje elveszik (lásd. 7 előadás)
  - 12 utasításciklus a megszakítási szubrutin

    → 15*1000*62.5ns = 937,5 µs-ig foglalt a processzor

Megszakítások és kihasználtság

- A teljes adatküldés 520ms
- A processzor ezzel összesen 937,5 µs-ig foglalkozik.

- Azaz a processzor „foglaltsága”: 0,18%

- Az idő 99,82%-ában bármi mást tehetünk.

- Lehetne-e jobban is?

DMA

Az adatmásolás gyakran ismétlődő feladat

- Megoldás #1: szoftveres ciklus → már láttuk
- Megoldás #2: megszakítás → már láttuk
- Megoldás #3: hardveres „másoló periféria”

→DMA vezérlő

DMA: Direct Memory Access

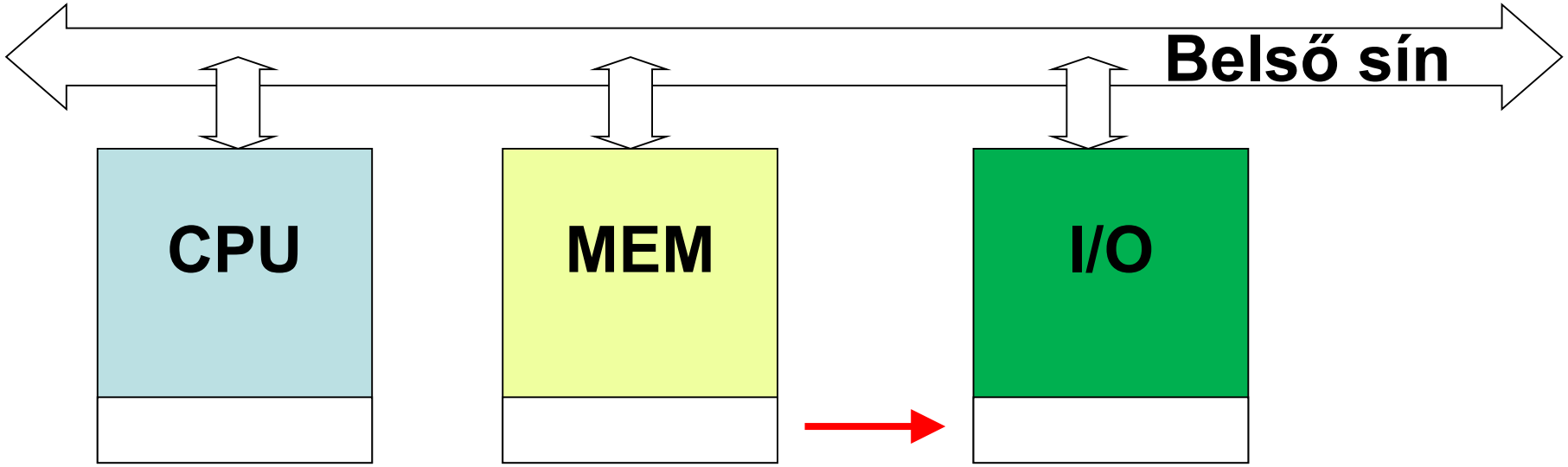
Adatmozgatás DMA nélkül

Résztvevők: CPU, Memória, Periféria

Pl.: Memória → I/O közötti átvitel

Ki irányít? →CPU

Egy Master, 2 slave



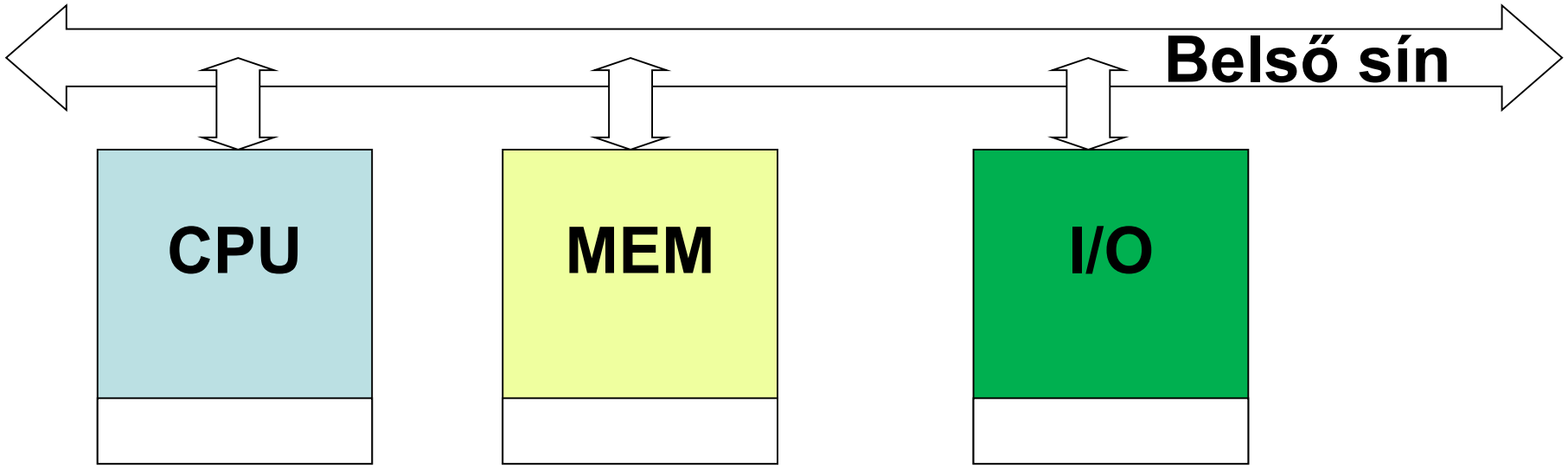
Adatmozgatás DMA nélkül

Résztvevők: CPU, Memória, Periféria

Pl.: Memória → I/O közötti átvitel

Ki irányít? →CPU

1. lépés: CPU kiolvassa az adatot a memóriából



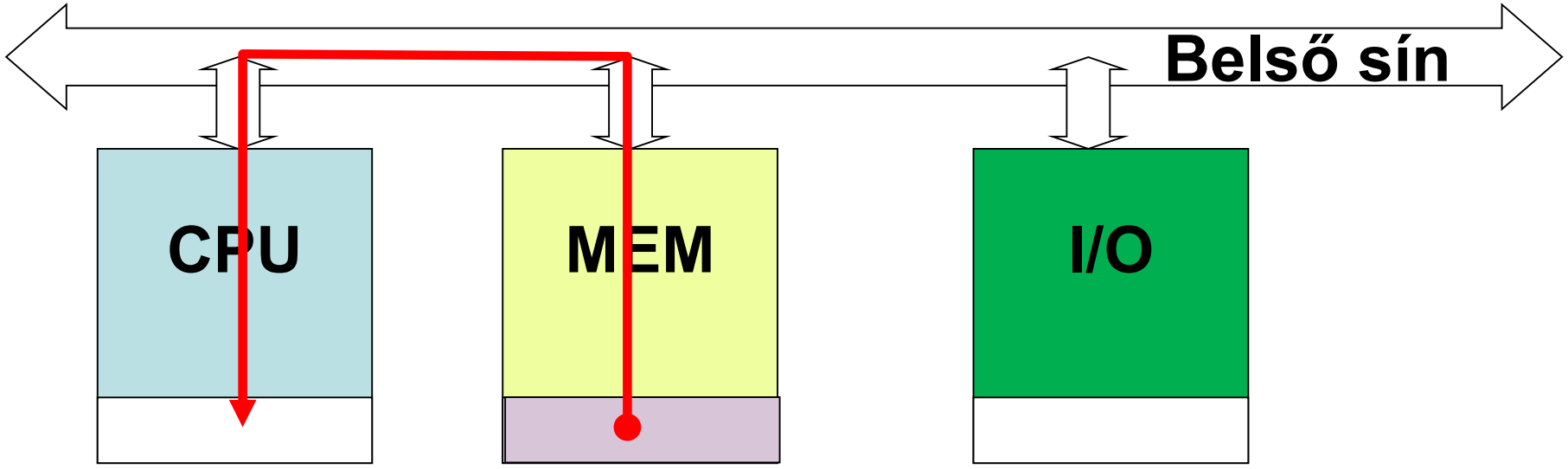
Adatmozgatás DMA nélkül

Résztvevők: CPU, Memória, Periféria

Pl.: Memória → I/O közötti átvitel

Ki irányít? →CPU

1. lépés: CPU kiolvassa az adatot a memóriából




```
mov.b kuldoptr,WREG
```

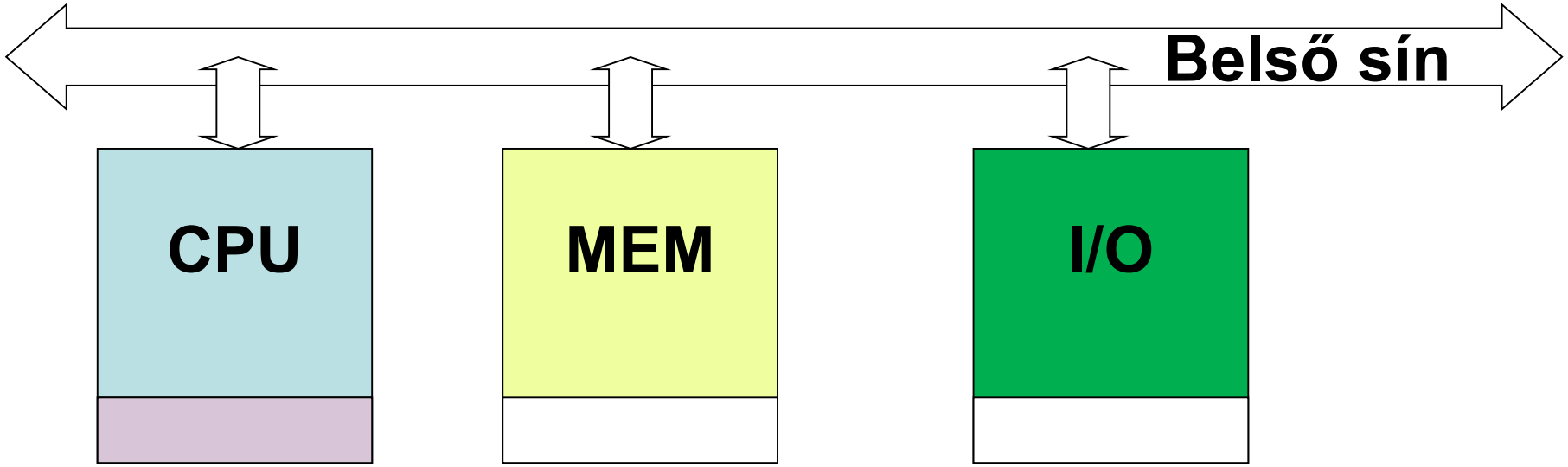
Adatmozgatás DMA nélkül

Résztvevők: CPU, Memória, Periféria

Pl.: Memória → I/O közötti átvitel

Ki irányít? →CPU

2. lépés: CPU beírja az adatot a perifériába



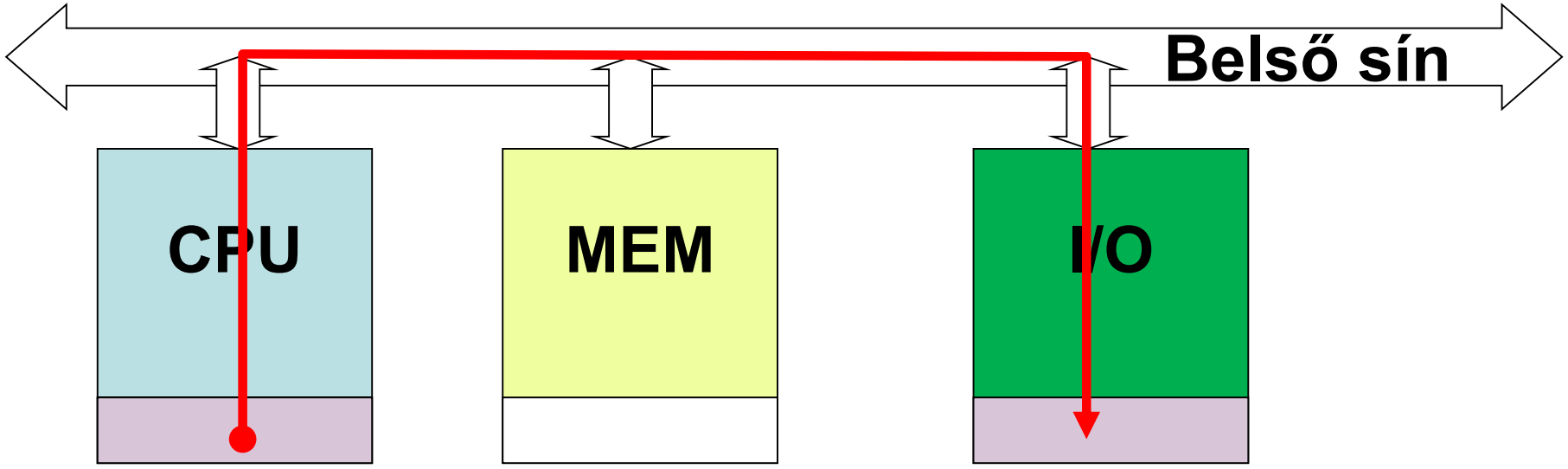
Adatmozgatás DMA nélkül

Résztvevők: CPU, Memória, Periféria

Pl.: Memória → I/O közötti átvitel

Ki irányít? →CPU

2. lépés: CPU beírja az adatot a perifériába



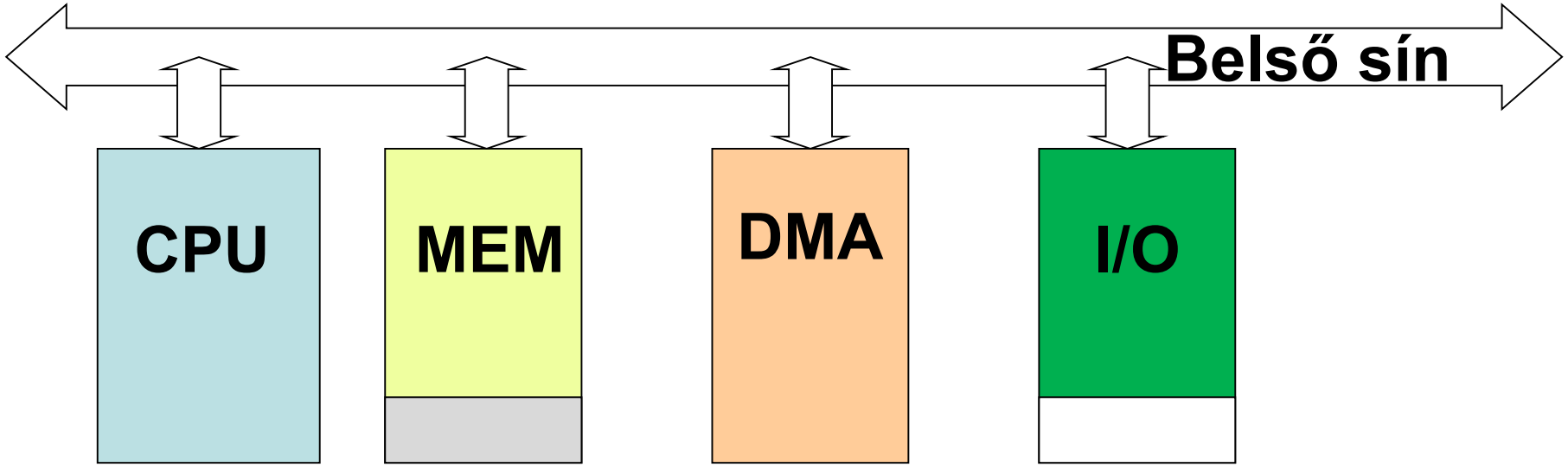

```
mov.b WREG,U1TXREG
```

Átvitel DMA vezérlővel

Résztvevők: CPU, Memória, Periféria, DMA

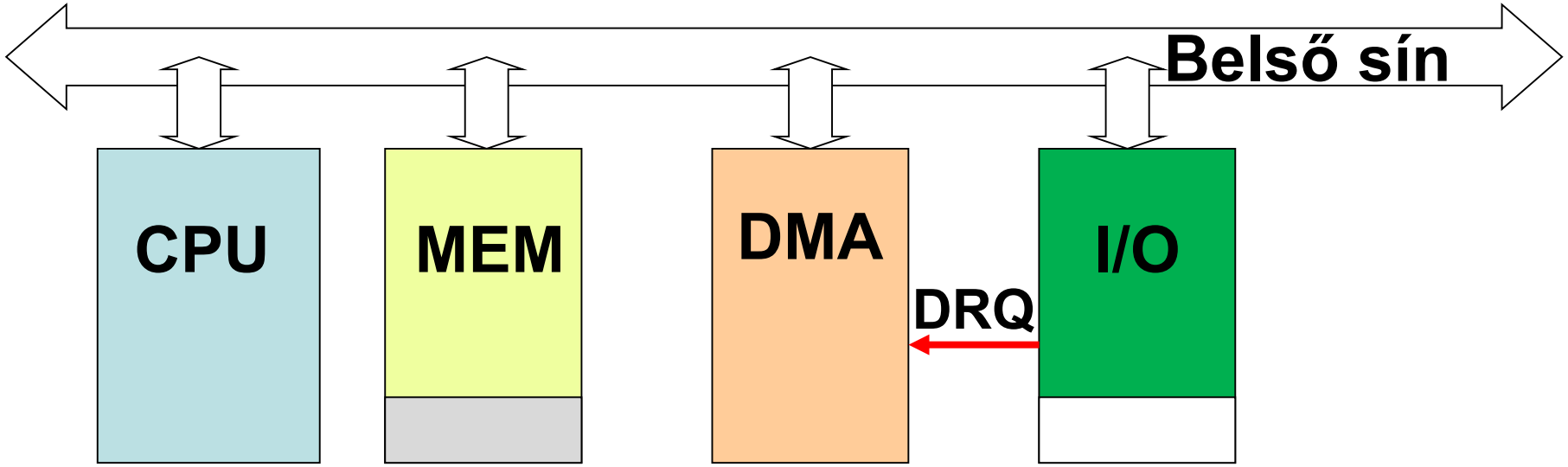
Ki irányít? →DMA

CPU és DMA lehet master→kié a sín?



Átvitel DMA vezérlővel

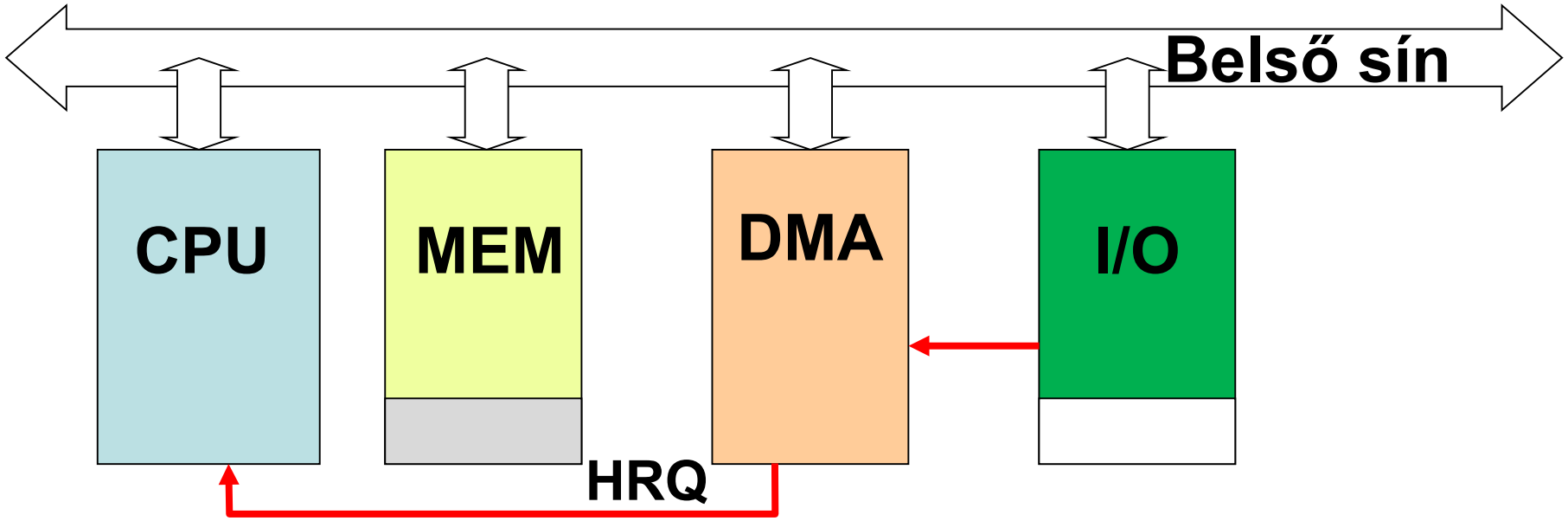
- Felprogramozás után a periféria jelzi átviteli igényét



DRQ: DMA ReQuest

Átvitel DMA vezérlővel

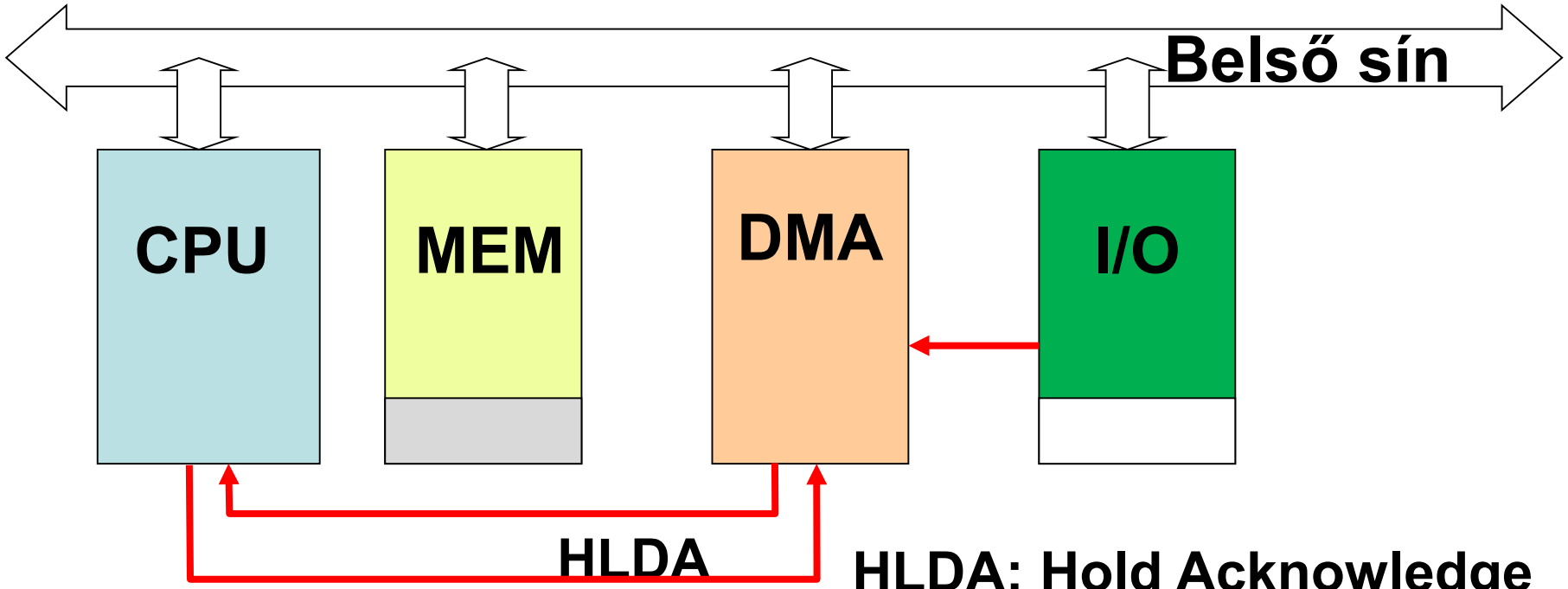
- DMA sínhozzáférést kér a CPU-tól



HRQ: Hold ReQuest

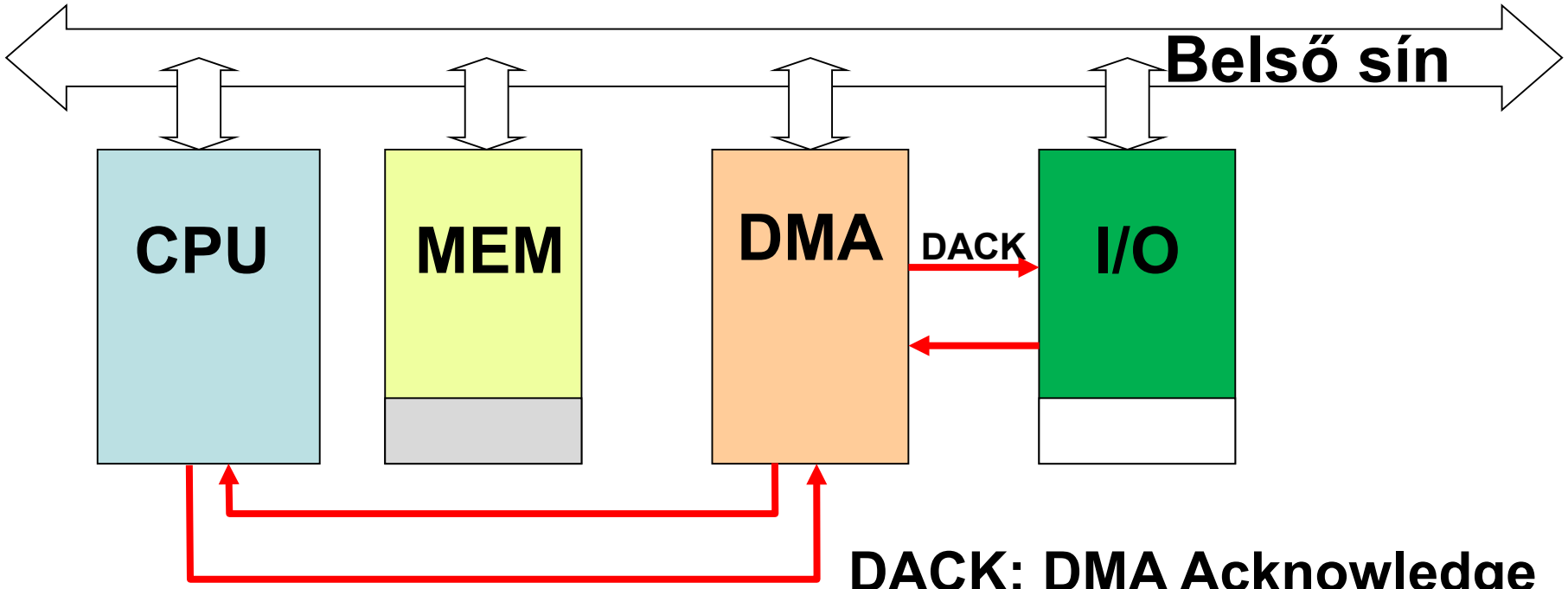
Átvitel DMA vezérlővel

- DMA sínhozzáférést kap a CPU-tól, a CPU átengedi a sín vezérlését a DMA vezérlőnek



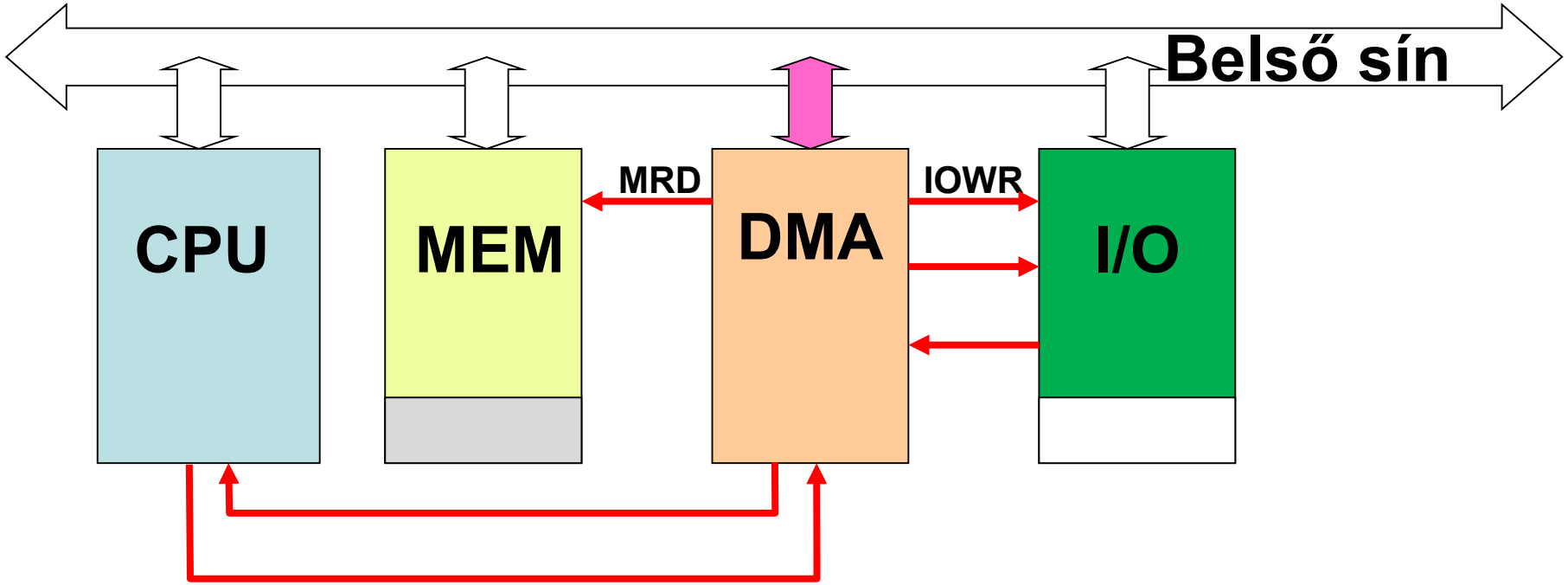
Átvitel DMA vezérlővel

- DMA nyugtázza az átviteli kérést (kiválasztja az I/O egységet)



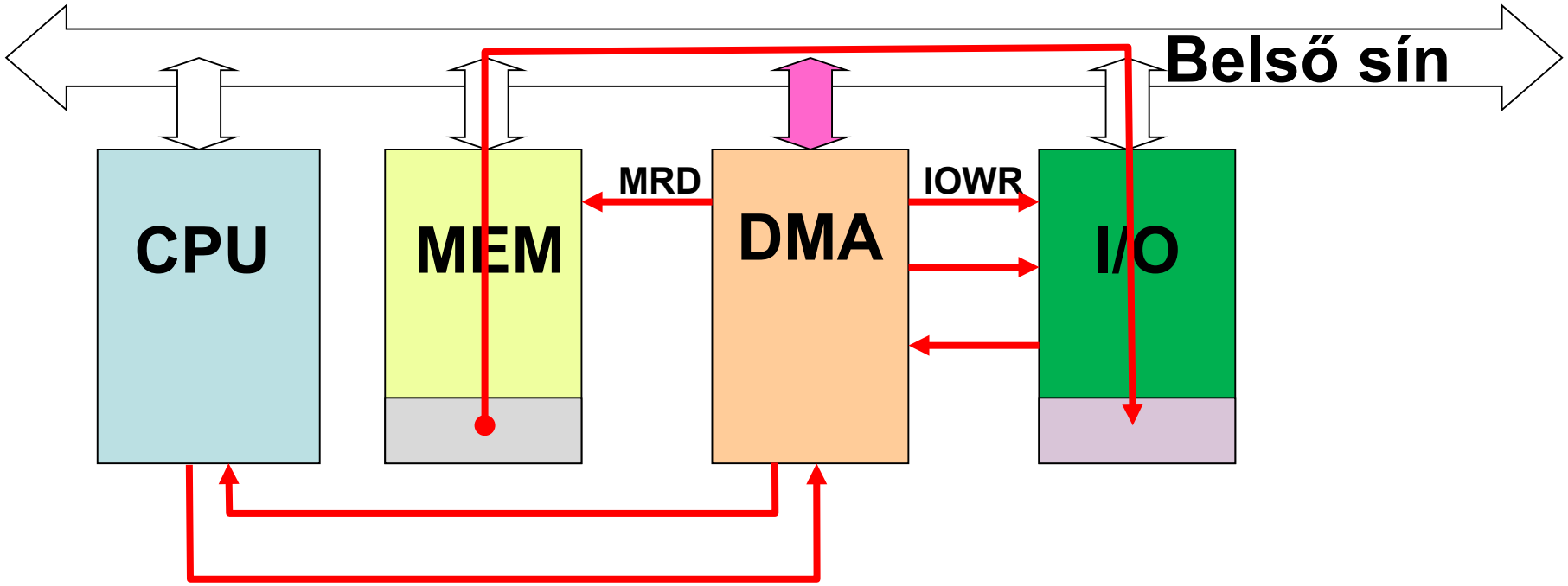
Átvitel DMA vezérlővel

- DMA írást jelez az I/O-nak és olvasást jelez a MEM-nek (egyszerre)
- A sínen memóriacím van



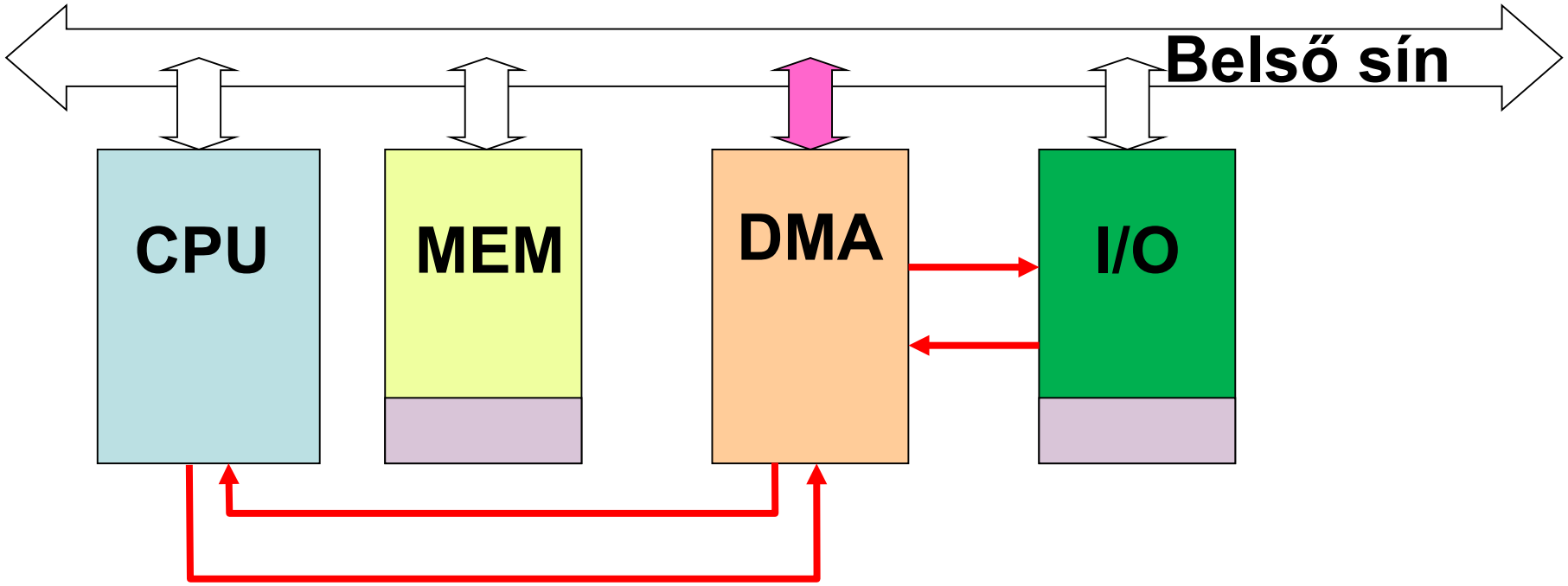
Átvitel DMA vezérlővel

- Az adat közvetlenül a memóriából a perifériára kerül



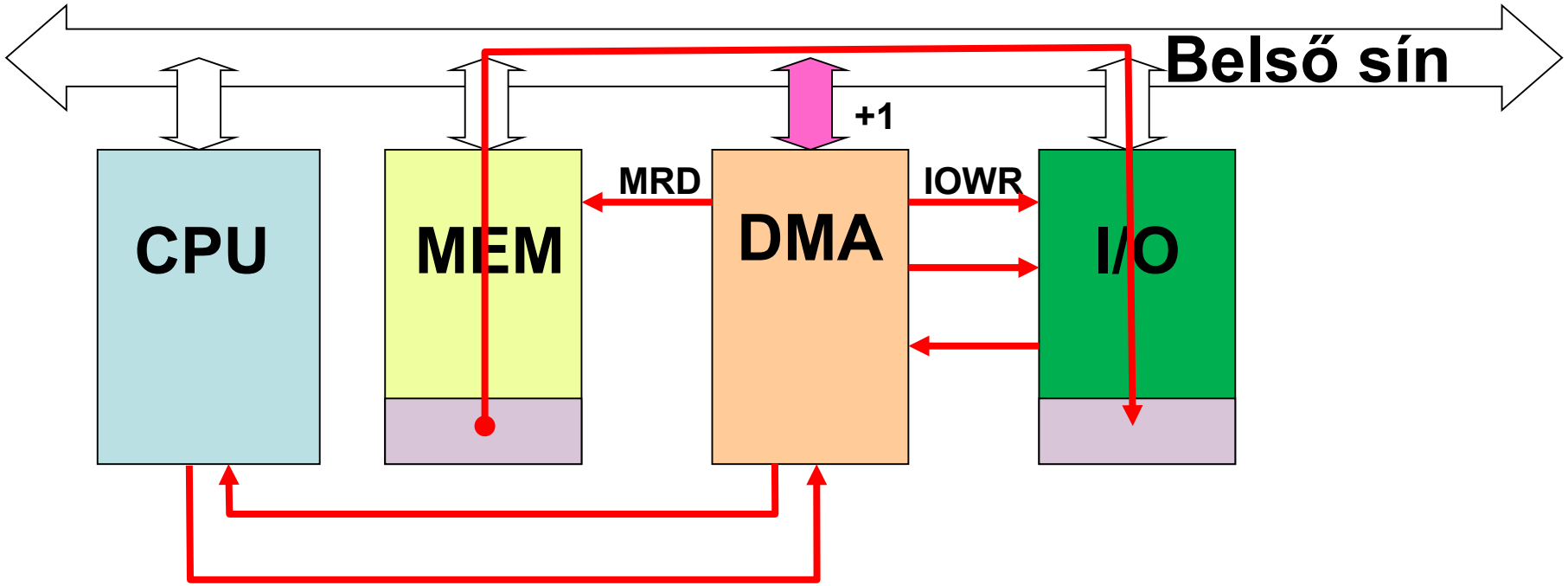
Átvitel DMA vezérlővel

- Ha további igény is van, a DMA nem engedi el a sínt



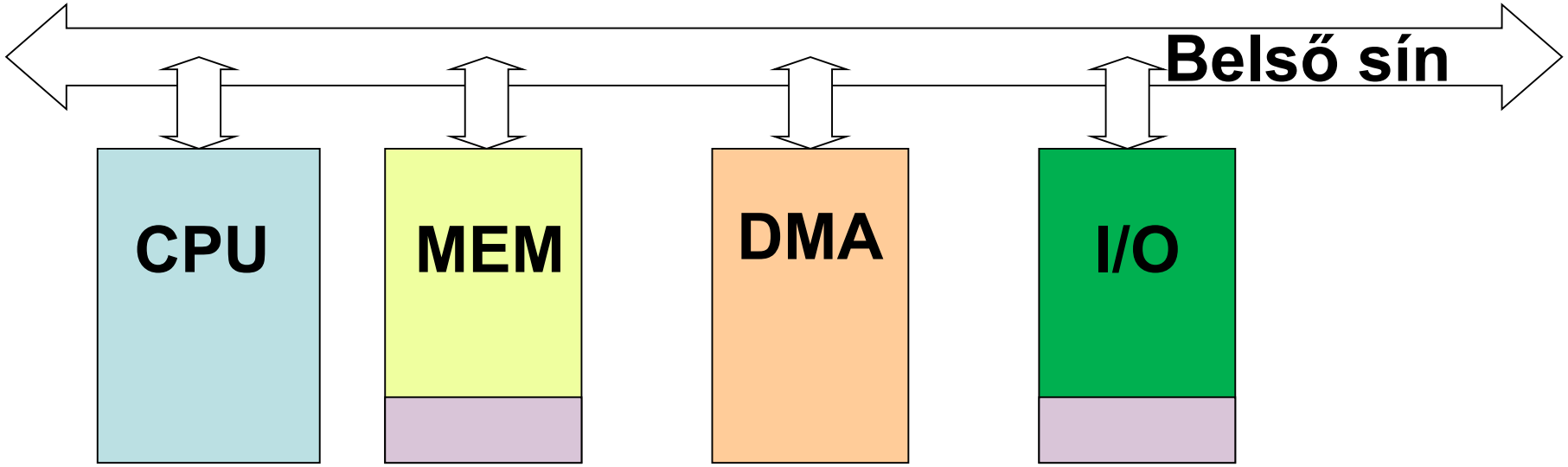
Átvitel DMA vezérlővel

- Folytatódik a másolás a növelt címmel (BURST mód)



Átvitel DMA vezérlővel

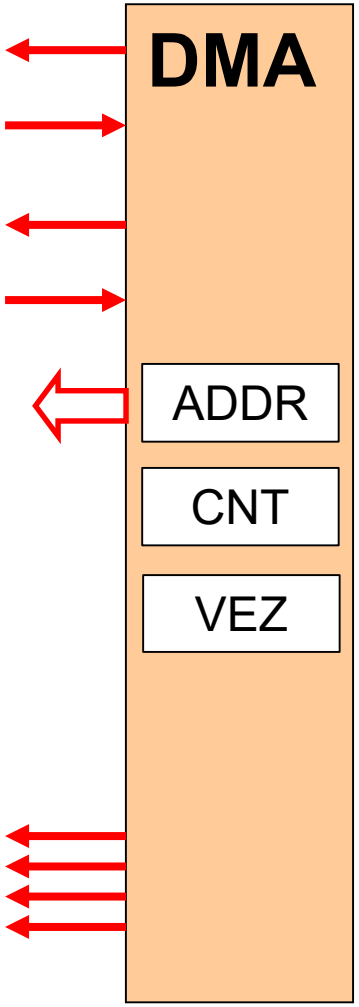
- Ha nincs több igény, vagy elértük a kért darabszámot, akkor a DMA elengedi a sínt



Erőforrás igény (DMA)

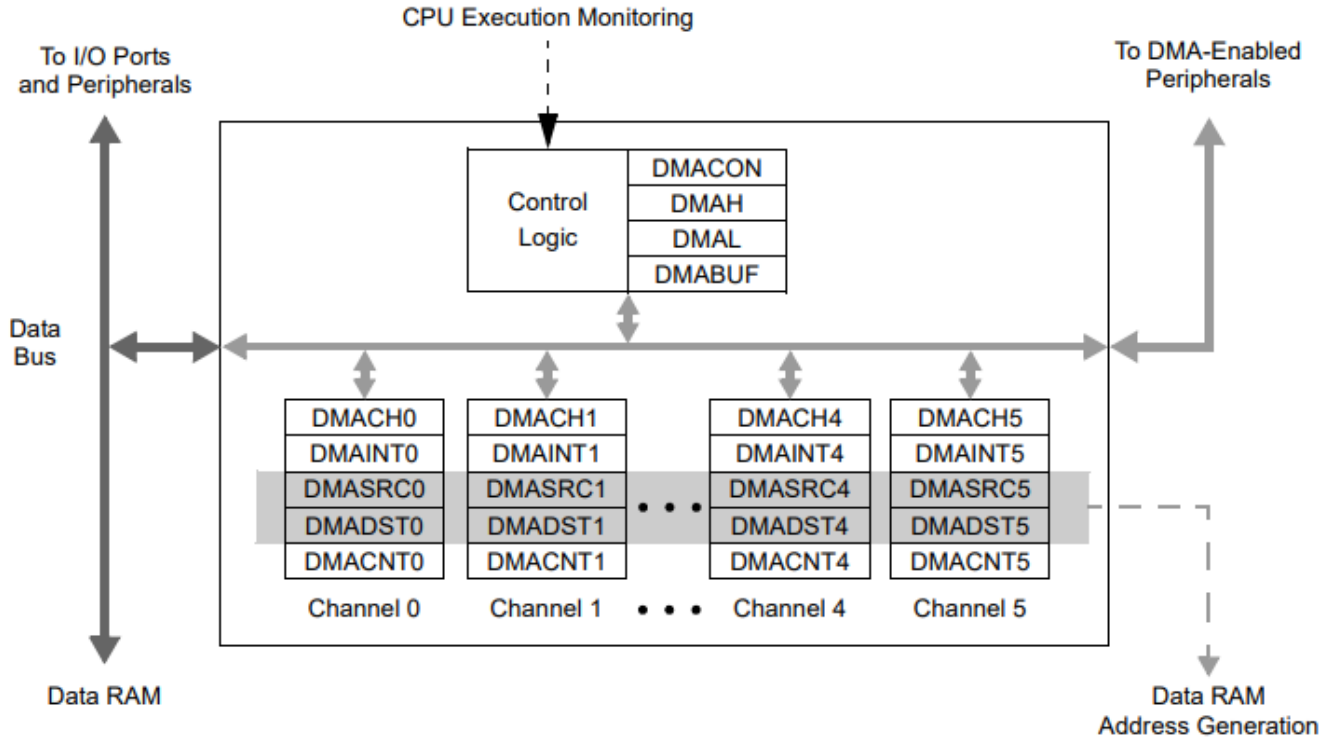
- Külön jelzővezetékek perifériákhoz (DRQ,DACK)
- Külön jelzővezetékek CPU-hoz (HRQ,HLDA)
- Csatornánként egy címszámláló
- Csatornánként egy darabszámláló
- Vezérlő/Jelzőbitek a működés engedélyezéséhez/tiltásához
- Sínvezérlés jelei

  (IORD,IOWR, MRD, MWR)

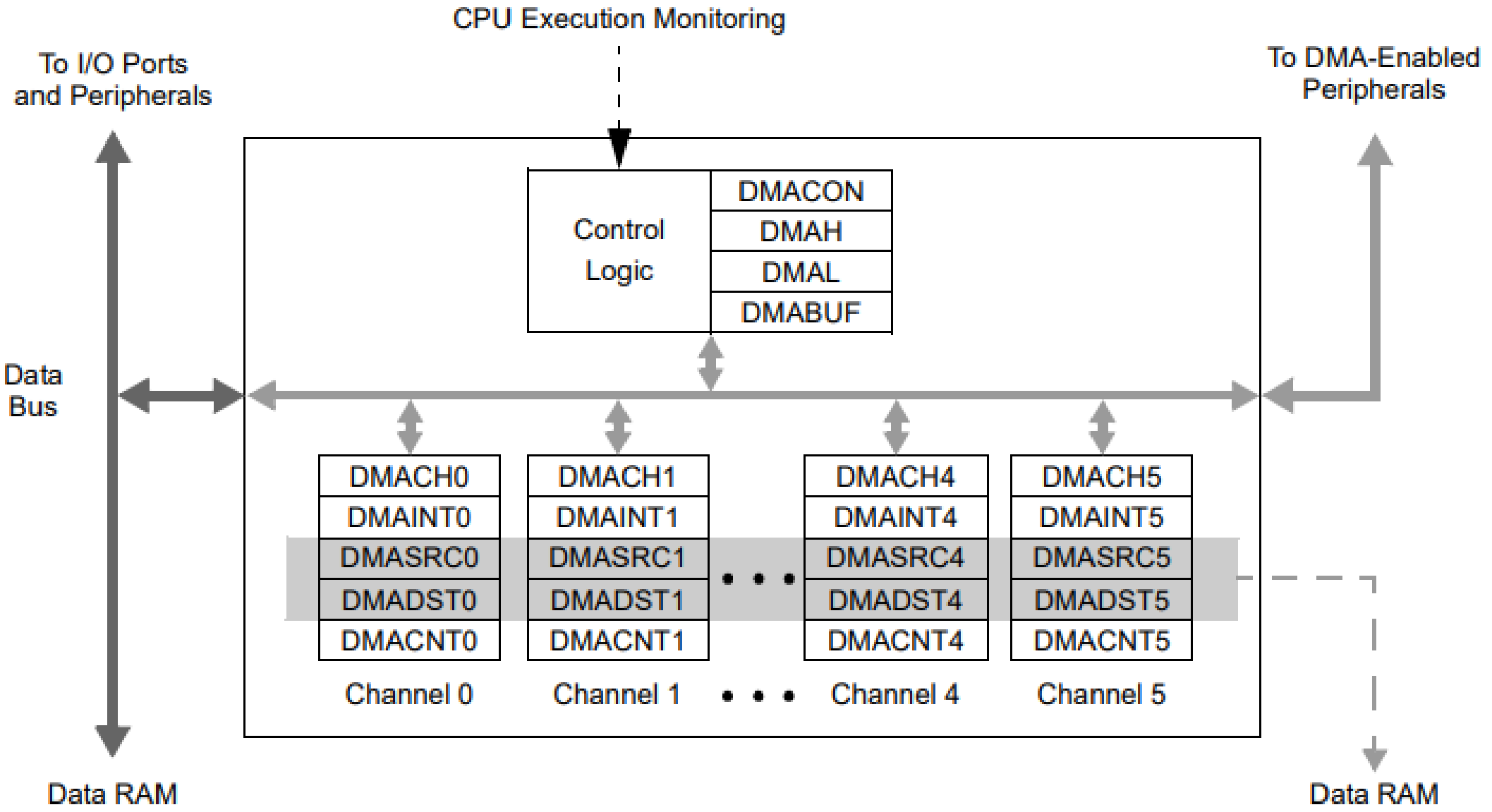


DMA a PIC24-ben

- A CPU-val teljesen párhuzamosan képes működni.
- A megszakítás vonalakat használja a periféria jelzéseiként
- 2-8 független csatorna →most éppen 6
- Minden csatornához tartozik:
  - Vezérlő regiszter (DMACHn)
  - Indítójel kiválasztás (DMAINTn)
  - Forrás pointer (DMASRCn)
  - Cél pointer (DMADSTn)
  - Darabszám (DMACNTn)



DMA a PIC24-ben



DMA globális vezérlő regiszter

- Csak 2 bit:
  - Be/kikapcsolás
    - Tejesen kikapcsolja az eszközt

    →A csatornákat külön-külön nem lehet kapcsolgatni, ameddig ez nincs bekapcsolva
  - Prioritás választás:
    - Egyszerre csak egy csatorna képes adatátvitelre.
    - Ha többnek is szükség lenne a működésére egy időben, akkor köztük prioritás-logika dönt a sorrendről.

REGISTER 5-1: DMACON: DMA ENGINE CONTROL REGISTER

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
DMAEN	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	PRSSEL
bit 7							bit 0

DMA csatornák prioritása

- Kétféle választható prioritás séma:
  - Fix prioritás (PRSSEL=0)
    - Mindig a kisebb sorszámú csatorna „nyer”, ha több egyidejű kérés van
    - Ha pl a 0. sorszámúnak „állandóan” jön kérés, a többi sosem „jut szóhoz”
    - A kisebb prioritású (nagyobb sorszámú) csatornák „kiéheztethetők”
  - Forgatott (Round-robin) prioritás (PRSSEL=1)
    - A csatornák prioritása kezdetben a sorszámuk szerinti.
    - Amikor egy csatorna átvitelt végez, a prioritása automatikusan a lehető legalacsonyabbra áll. („ a prioritások átfordulnak”)
      → Idővel az összes idővel szóhoz tud jutni, nincs „kiéhezés”

DMA csatornák prioritása

- Forgatott (Round-robin) prioritás (PRSSEL=1)
  - A csatornák prioritása kezdetben a sorszámuk szerinti (0 a legmagasabb)
  - Amikor egy csatorna átvitelt végez, a prioritása automatikusan a lehető legalacsonyabbra áll. („ a prioritások átfordulnak”)
    → Idővel az összes szóhoz tud jutni, nincs „kiéhezés”

Table 5-1: Examples of Channel Access Using Round-Robin Priority Scheme

DMA-t kérő csatornák				A lehetőséget megkapja
0	**1**	**2**	**3**	
prioritás				
0	1	2	3	None
2	3 X	0	1	CH1
1 X	2 X	3 X	0	CH2
3 X	0 X	1	2	CH0
2 X	3 X	0	1	CH1
0 X	1 X	2	3 X	CH3
3 X	0 X	1	2	CH0

DMA csatorna vezérlő regiszter

- Fontosabb lehetőségek:
  - CHEN: ki/bekapcsolás
  - SIZE: 8 bites (1) vagy 16 bites (0) mód
  - TRMODE: átvitel módja
  - DAMODE: cél cím módosítás módja
  - SAMODE: forrás cím módosítás módja
  - RELOAD: A tranzakció végén újratölti a kezdeti értékeket
  - CHREQ: DMA kérés „kézzel”

REGISTER 5-2: DMACHn: DMA CHANNEL n CONTROL REGISTER

U-0	U-0	U-0	r-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	NULLW	RELOAD(1)	CHREQ(3)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SAMODE1	SAMODE0	DAMODE1	DAMODE0	TRMODE1	TRMODE0	SIZE	CHEN
bit 7							bit 0

DMA csatorna üzemmódjai

- A TRMODE lehetséges értékeitől függően:

  00: One-Shot mode
  - Egy indítás után egy tranzakciót végez el
  - A darabszám regisztert csökkenti
  - Ha a darabszám regiszter 0-ra csökken, akkor megszakítást kér, megáll (CHEN=0-ra áll, letiltja magát).

  01: Repeated One-Shot mode
  - Egy indítás után egy tranzakciót végez
  - A darabszám regisztert csökkenti
  - Ha a darabszám regiszter 0-ra csökken, visszaállítja az eredeti értékeket (kezdőcímekre és hosszra), és ezután újra indítható.
  - Megszakítást nem kér

DMA csatorna üzemmódjai

10 = Continuous mode

- Egy indítás esetén a darabszám regiszterben megadott darabszámú tranzakciót végez
- Az utolsó tranzakció végén megáll és megszakítást kér.

11 = Repeated Continuous mode

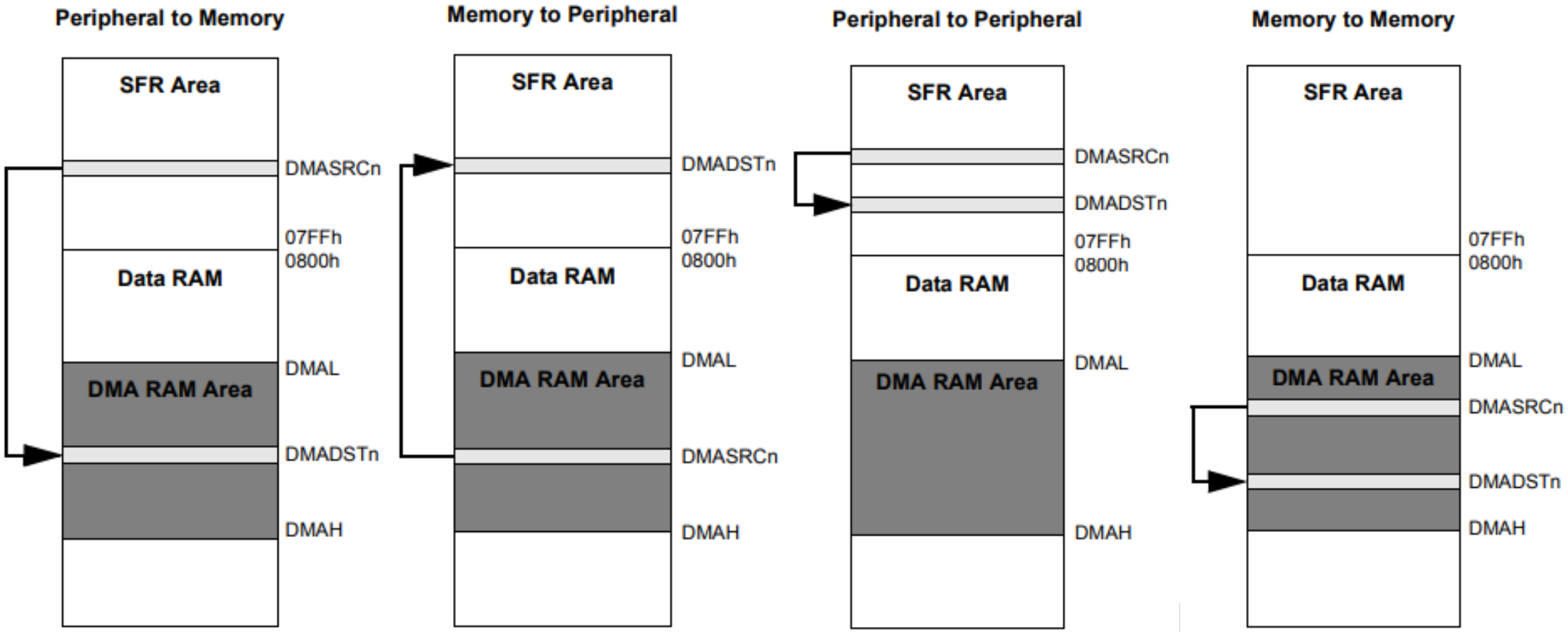
- Minden indítás esetén a darabszám regiszterben megadott darabszámú tranzakciót végez.
- Az utolsó tranzakció végén visszaállítja az eredeti értékeket, és újraindítható, nem kér megszakítást sem.

Cím módosítási lehetőségek

- DAMODE/SAMODE
  - Cél/forrás cím módosítás lehetőségei:
    - 00: adott cím nem változik
    - 01: adott cím növekszik (1-esével vagy 2-esével az adatmérettől függően)
    - 10: adott cím csökken (1-esével vagy 2-esével az adatmérettől függően)
    - 11: az adott címet (vagy egy részét) a periféria szolgáltatja
      - » Vannak perifériák, amelyek képesek címet generálni, jellemzően ilyenek, amik egyben nagy adatblokkokat vagy struktúrákat adnak át DMA-val, ilyenkor képesek a közvetlenül saját pufferükben csak részeket átmásolni
      - » Nem tanulunk ilyent *(Ezen a kontrolleren ilyen csak az AD átalakító, más típusokon ilyen szokott lenni az USB és a CAN vezérlő)*

Lehetséges adatmozgatások

- Az összes lehetséges irányba képes adatot mozgatni.
- DAMODE/SAMODE megfelelő megadása kell hozzá:



Indítójel kiválasztása

- A DMA csatornák valamelyik megszakítás vonal hatására végzik működésüket.
- Az adott periféria megszakítás kérése indítja a DMA tranzakciót, a tranzakció végén a DMA törli is a megszakítás flaget.

REGISTER 5-3: DMAINTn: DMA CHANNEL n INTERRUPT REGISTER

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DBUFWF(1)	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
HIGHIF(1,2)	LOWIF(1,2)	DONEIF(1)	HALFIF(1)	OVRUNIF(1)	—	—	HALFEN
bit 7							bit 0

DMA indítójel kiválasztása

REGISTER 5-3: DMAINTn: DMA CHANNEL n INTERRUPT REGISTER

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DBUFWF(1)	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
HIGHIF(1,2)	LOWIF(1,2)	DONEIF(1)	HALFIF(1)	OVRUNIF(1)	—	—	HALFEN
bit 7							bit 0

TABLE 5-1: DMA TRIGGER SOURCES

CHSEL[6:0]	Trigger (Interrupt)	CHSEL[6:0]	Trigger (Interrupt)
0000000	Off	1000001	UART2 TX Interrupt
0001001	MCCP4 IC/OC Interrupt	1000010	UART2 RX Interrupt
0001010	MCCP4 Timer Interrupt	1000011	UART2 Error Interrupt
0001011	MCCP3 IC/OC Interrupt	1000100	UART1 TX Interrupt
0001100	MCCP3 Timer Interrupt	1000101	UART1 RX Interrupt
0001101	MCCP2 IC/OC Interrupt	1000110	UART1 Error Interrupt
0001110	MCCP2 Timer Interrupt	1001011	DMA Channel 5 Interrupt
0001111	MCCP1 IC/OC Interrupt	1001100	DMA Channel 4 Interrupt
0010000	MCCP1 Timer Interrupt	1001101	DMA Channel 3 Interrupt
0010100	OC3 Interrupt	1001110	DMA Channel 2 Interrupt
0010101	OC2 Interrupt	1001111	DMA Channel 1 Interrupt
0010110	OC1 Interrupt	1010000	DMA Channel 0 Interrupt
0011010	IC3 Interrupt	1010001	A/D Interrupt
0011011	IC2 Interrupt	1010011	PMP Interrupt
0011100	IC1 Interrupt	1010100	HLVD Interrupt
0100000	SPI3 Receive Interrupt	1010101	CRC Interrupt

Az átvitel végének ellenőrzése

- DMAINTn regiszter flagjei mutatják az állapotot:
  - DONEIF: Átvitel teljesen elkészült (legalább egyszer)
  - HALFIF: Az átvitel már a feléig elkészült (DMACNTn regiszter a fele az eredeti értékének)
  - OVRUNIF: Új indító jel érkezett, miközben az előző tranzakció nem volt még kész

REGISTER 5-3: DMAINTn: DMA CHANNEL n INTERRUPT REGISTER

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DBUFWF(1)	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
HIGHIF(1,2)	LOWIF(1,2)	DONEIF(1)	HALFIF(1)	OVRUNIF(1)	—	—	HALFEN
bit 7							bit 0

Használjuk a DMA-t az UART adásra

Lépések:

1. Inicializáljuk a DMA-t
2. Elindítjuk az első kérést

→ Innentől az UART kéri majd a többi DMA-t, és az elintézi a többi küldést.

Ha figyelni akarjuk, hogy elkészült-e, nézhetjük a DONEIF flaget az adott DMA csatornában:

```
var:  btst DMAINT0,#DONEIF
      bra z,var
```

(vagy megszakítást is készíthetünk rá)

DMACH0 beállítása

- Szükséges beállítások:
  - CHEN: majd akkor kapcsoljuk, ha kell küldeni!
  - SIZE: adatméret - 8 bit (1)
  - TRMODE: one shot (00)
  - DAMODE: cél címe az U1TXREG, azt ne módosítsuk (00)
  - SAMODE: forrás cím növekedjen (01)

REGISTER 5-2: DMACHn: DMA CHANNEL n CONTROL REGISTER

U-0	U-0	U-0	r-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	NULLW	RELOAD(1)	CHREQ(3)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SAMODE1	SAMODE0	DAMODE1	DAMODE0	TRMODE1	TRMODE0	SIZE	CHEN
bit 7							bit 0

0x0042

DMA indítójel kiválasztása

REGISTER 5-3: DMAINTn: DMA CHANNEL n INTERRUPT REGISTER

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DBUFWF(1)	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
HIGHIF(1,2)	LOWIF(1,2)	DONEIF(1)	HALFIF(1)	OVRUNIF(1)	—	—	HALFEN
bit 7							bit 0

0x4400

TABLE 5-1: DMA TRIGGER SOURCES

CHSEL[6:0]	Trigger (Interrupt)	CHSEL[6:0]	Trigger (Interrupt)
0000000	Off	1000001	UART2 TX Interrupt
0001001	MCCP4 IC/OC Interrupt	1000010	UART2 RX Interrupt
0001010	MCCP4 Timer Interrupt	1000011	UART2 Error Interrupt
0001011	MCCP3 IC/OC Interrupt	1000100	UART1 TX Interrupt
0001100	MCCP3 Timer Interrupt	1000101	UART1 RX Interrupt
0001101	MCCP2 IC/OC Interrupt	1000110	UART1 Error Interrupt
0001110	MCCP2 Timer Interrupt	1001011	DMA Channel 5 Interrupt
0001111	MCCP1 IC/OC Interrupt	1001100	DMA Channel 4 Interrupt
0010000	MCCP1 Timer Interrupt	1001101	DMA Channel 3 Interrupt
0010100	OC3 Interrupt	1001110	DMA Channel 2 Interrupt
0010101	OC2 Interrupt	1001111	DMA Channel 1 Interrupt
0010110	OC1 Interrupt	1010000	DMA Channel 0 Interrupt
0011010	IC3 Interrupt	1010001	A/D Interrupt
0011011	IC2 Interrupt	1010011	PMP Interrupt
0011100	IC1 Interrupt	1010100	HLVD Interrupt
0100000	SPI3 Receive Interrupt	1010101	CRC Interrupt

Küldés DMA-val

```
Uart_send_array_DMA:
    mov #0x0042,w0 ;CH0 beállítás: One shot mód, célcím marad, forráscím nő, bájtos mód
    mov w0,DMACH0
    mov #0x4400,w0 ;a 0x44-es IT-re indul, az az UART1 TX interrupt vonala
    mov w0,DMAINT0
    mov #tomb,w0
    mov w0,DMASRC0  ;a DMA CH0 forráscíme, tomb
    mov #U1TXREG,w0 ;itt a # nagyon kell - a TXREG címére van szükségünk
    mov w0,DMADST0  ;a DMA CH0 célcíme, U1TXREG
    mov #1000,w0
    mov w0,DMACNT0 ;1000 bájtot másolunk

    bset DMACON,#DMAEN_DMACON ;DMA globálisan bekapcsol
    bset DMACH0,#CHEN ;DMA CH0 is bekapcsol

    bclr IFS0,#U1TXIF  ;már régóta be volt állítva a TXIF, erre nem fog indulni a DMA
    bset DMACH0,#CHREQ ;az első átvitelt mi indítjuk (az UART régóta kész)
    return
```

DMA és kihasználtság

- Csak az inicializáláshoz kell egy (most épp) 19 ciklusig tartó szubrutin.
- Utána semmilyen CPU használat nincs.
- A DMA átvitel néha 1-1 (összesen 1000) ciklusidőt igényel, de:
  - Az adott típuson a DMA képes teljesen a „háttérben” dolgozni, azaz amelyik fázisban a CPU épp nem használja a memóriát, akkor ő tudja használni.

    Elég sok ilyen van, mert ugye minden ciklusban 2 memóriahozzáférés történhetne, de nem mindig szokott.

  →Legjobb esetben konkrétan nem is foglal processzort az átvitel

  →De a legrosszabb esetben is az idő legalább 99,988%-ában mást csinálhatunk.

DMA összefoglalás

- A DMA egy ciklusú adatátvitelre képes I/O és MEM között
- Automatikus cím és darab számolás, külön időveszteség nélkül
- Gyors válasz a periféria átvitel igényére
- A lehető legnagyobb sebességű átvitel így oldható meg
- CPU-nak elég az átvitel végeztével jelezni, nem kell minden bájtra megszakítást kérni

  →Blokkos adatátvitel