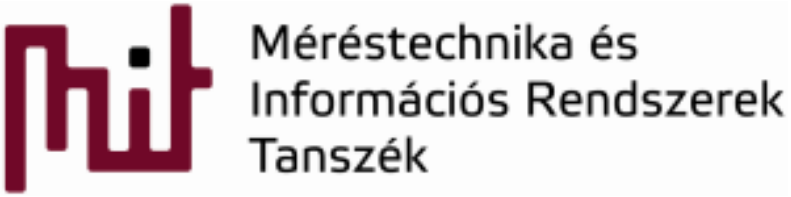
Nagyteljesítményű mikrovezérlők

7. NVIC

Scherer Balázs



Budapest University of Technology and Economics
Department of Measurement and Information Systems

ARM7, ARM9 megszakítás kezelés

- ARM7, ARM9 két interrupt vonal
  - IRQ: Normál priorítású IT
  - FIQ: Fast IT saját regiszter blokkal
  - A vektoros megszakításkezelés gyártó specifikus
  - Nem volt determinisztikus az interrupt kiszolgálás: attól függött a megszakítás kiszolgálása, hogy éppen milyen utasítás hajtódott végre.
  - Az ARM7, ARM9 hardware-esen nem támogatta az ún. Nested IT-ket. (IT-t megszakító IT)
- A Cortex M sorozat megszakítás kezelője a fenti korlátokra próbál megoldást adni.

Cortex M3 NVIC

- Nested Vector Interrupt Controller
  - Gyártó független standard tartozék, ebből következően gyártó független interrupt struktúra.
    - Könnyű portolhatóság
  - A Thumb2 utasításkészlet több órajelig tartó utasításai megszakíthatóak, így az IT kezelés determinisztikus.
  - Nested interuptokat támogatja
  - Bár az NVIC processzor független, az erőforrás használat minimalizálása miatt a processzor tervezők megszabhatják NVIC bemenő vonalainak számát.
    - Az NVIC képes: 1 nem maszkolható +240 külső periféria + 15 belső Cortex-es IT vonal forrást kezelni
      - Az STM32f107 43-at
      - Az LPC1768 35-öt használ

Az NVIC kezelése

- Meg kell adni az ugrótáblát és a prioritásokat.
- Az ugrótábla a címtartomány alján a 0x00000004-ről indul.
  - A 0x00000000-án a kezdő stack pointer van
    - Minél hamarabb lehessen C-t használni.
- Az első 15 megszakítás a Cortex Core-hoz tartozik
- Ezek után jönnek a gyártó specifikus periféria megszakítások

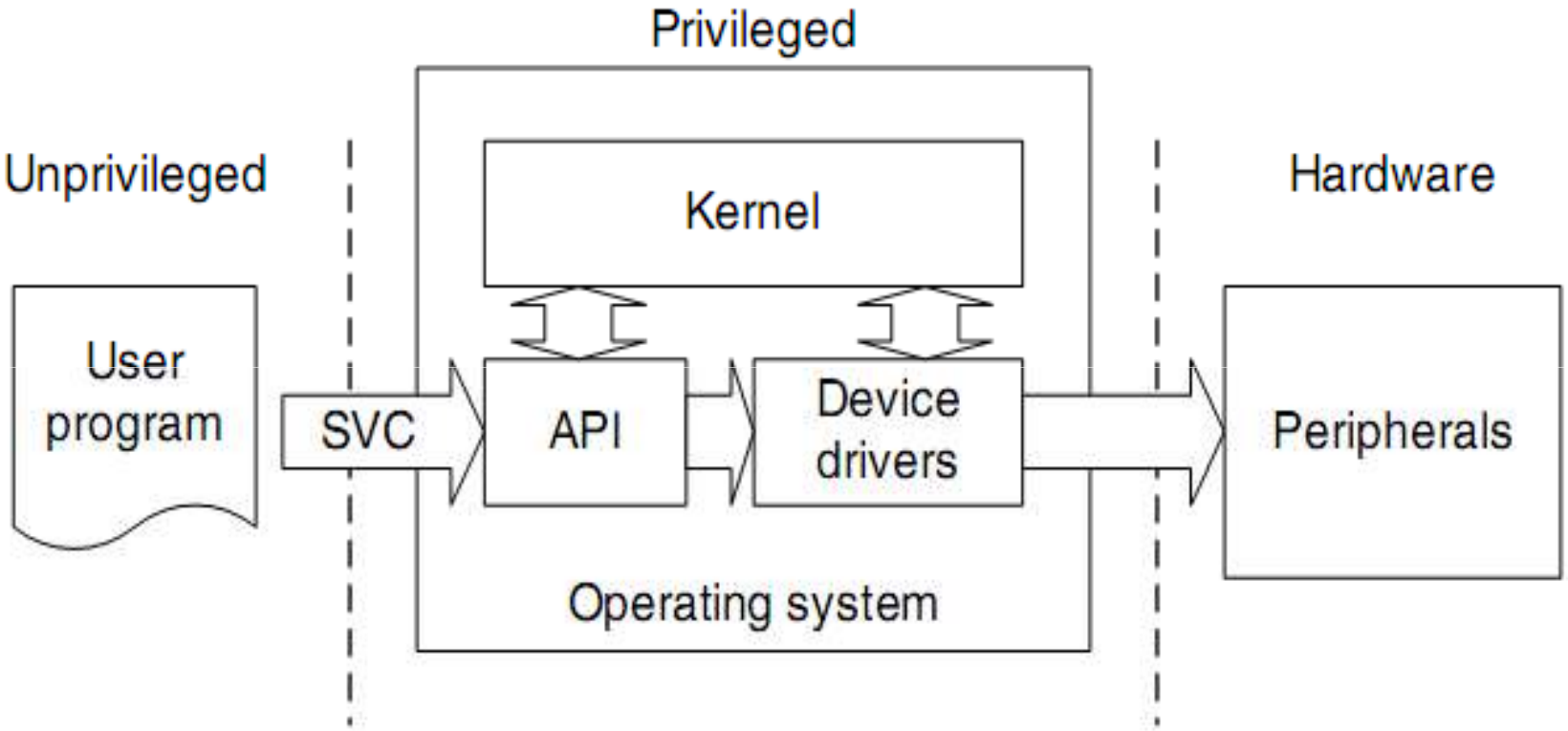
Az NVIC ugrótábla

- Az ugrótábla a címtartomány alján a 0x00000004-ről indul.
  - A 0x00000000-án a kezdő stack pointer van, hogy minél hamarabb lehessen C-t használni.

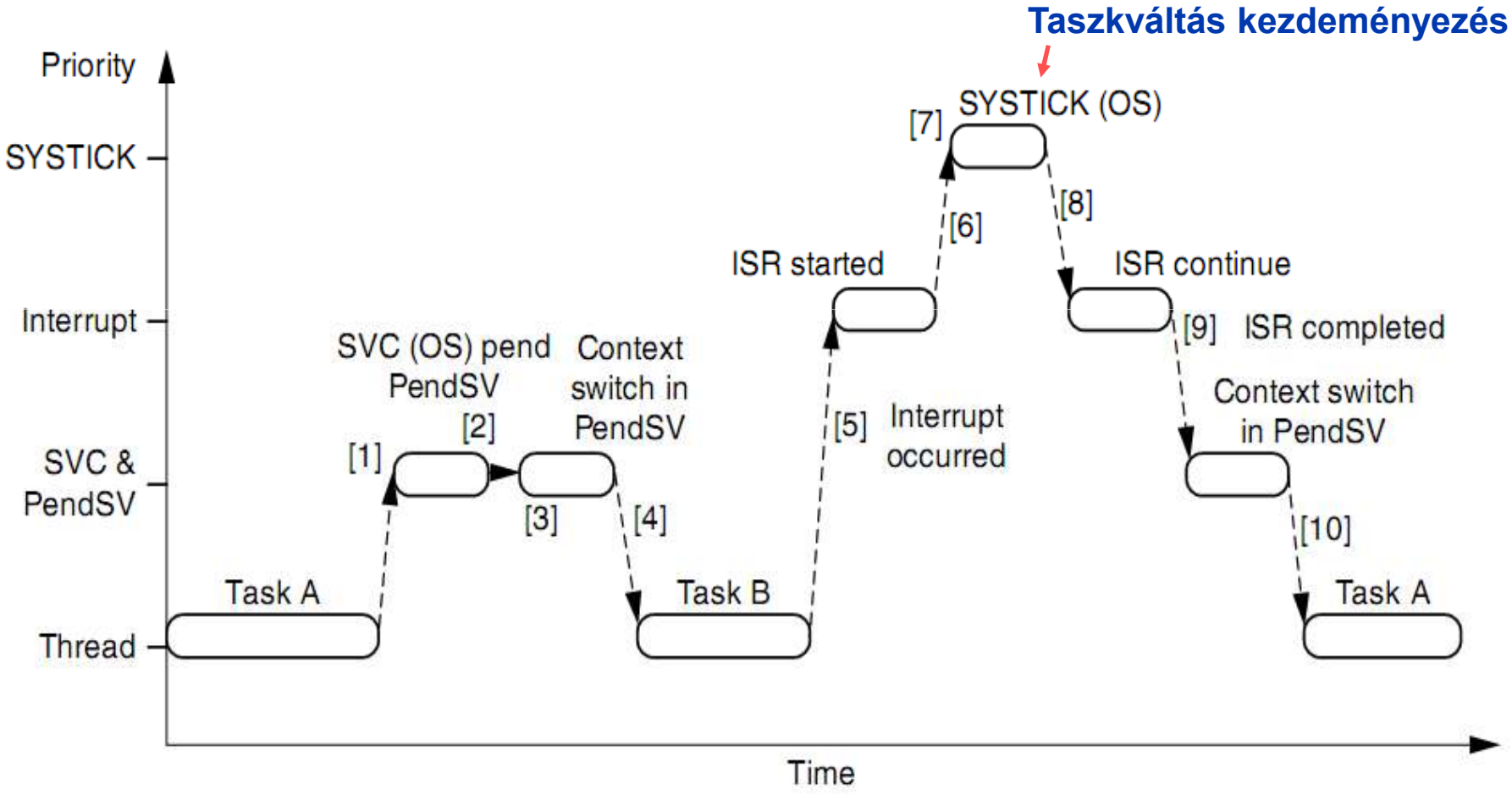
No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
.....	.....	.....	settable	.....
256	Interrupt#240	247	settable	External Interrupt #240

Gyártó specifikus

A SVC használata



A PendSV kezelése

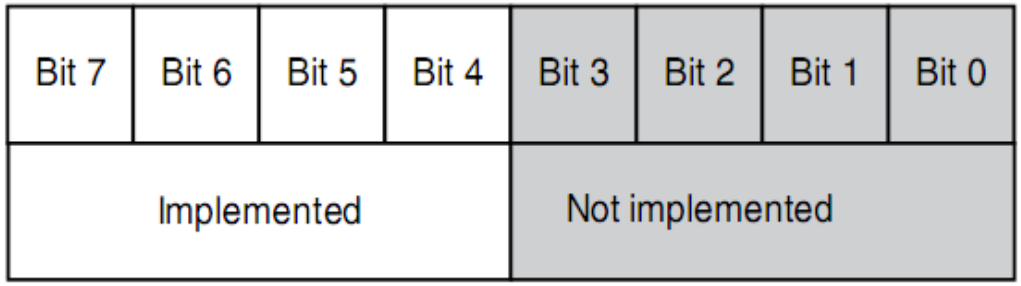


Alap regiszterek

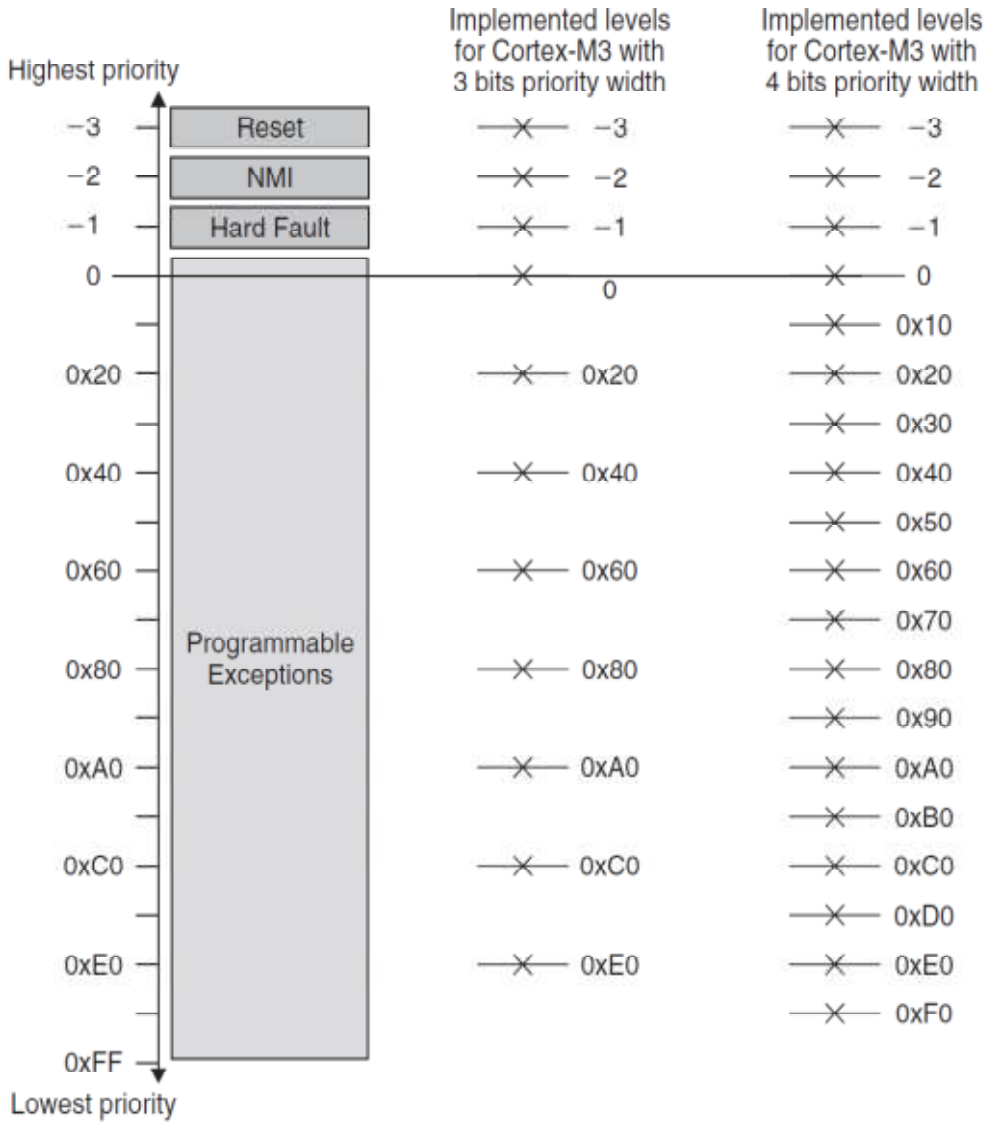
- **Interrupt enable** és **Clear enable** regiszterek
  - SETENA0-n/CLRENA0-n
    - 32bites külön tiltó és engedélyező regiszter
- **Interrupt set pending** és **Clear pending**
  - SETPEND0-n/CLRPEND0-n
    - 32bites regiszterek, amelyekből a megszakított, várakozó interruptokat lehet kiolvasni, törölni.

Prioritás kezelés

- Az alap belső kivételeknek fix prioritása van
- A többi külső megszakításhoz (a mag szempontjából) prioritás regiszter
  - max. 8 bites, min 3 bites prioritás regiszter
  - Korlátozni szokták a szinteket az egyszerűbb hardware kialakítás miatt
  - A MSB bitektől kezdődik az implementálás (könnyebb portolhatóság)
  - Az STM32F107, LPC1768, STM32F429 esetében 4 bites prioritás regi

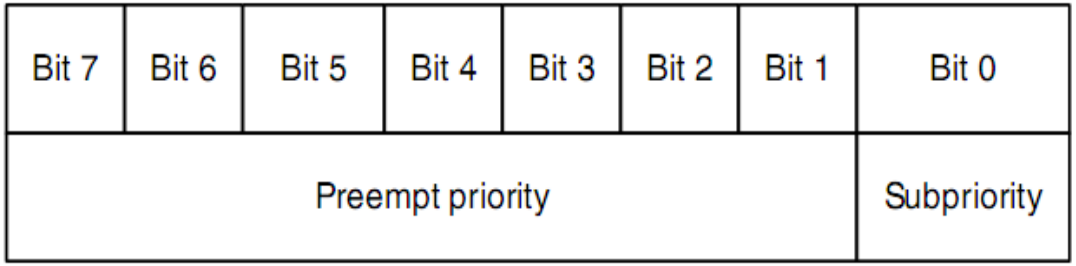


Prioritás kezelés *folytatás*



Preempt priority és Subpriority

- A 8 bit, de csak 127 preemptciós szint létezik
- Subprioritás
  - Azonos preemptciós szintű prioritások szerint az alacsonyabb subprioritású fut le először
  - PRIGROUP regiszter



Preempt priority és Subpriority

- A 8 bit, de csak 127 preemptciós szint létezik
- Subprioritás
  - Azonos preemptciós szintű prioritások szerint az alacsonyabb subprioritású fut le először
  - PRIGROUP regiszter
  - Az STM32F107, LPC1768, STM32F429 esetében 4 bites prioritás regiszter

PRIGROUP (3 Bits)	Binary Point (group.sub)		Preempting Priority (Group Priority)		Sub-Priority	
			Bits	Levels	Bits	Levels
011	4.0	gggg	4	16	0	0
100	3.1	gggs	3	8	1	2
101	2.2	ggss	2	4	2	4
110	1.3	gsss	1	2	3	8
111	0.4	ssss	0	0	4	16

További NVIC regiszterek, opciók

- Megszakítás maszkot
  - PRIMASK mindet kivéve a hibákat
  - FAULTMASK a hibákat is maszkolja -1 ig
  - BASEPRI egy bizonyos prioritás alatt maszkol
- Vector Table Offset Regiszter
  - Áthelyezhető az IT táblázat szinte tetszőleges helyre
  - Boot-olás, Boot-loader segítő lépés

Table 7.7 Vector Table Offset Register (Address 0xE000ED08)

Bits	Name	Type	Reset Value	Description
29	TBLBASE	R/W	0	Table base in Code (0) or RAM (1)
28:7	TBLOFF	R/W	0	Table offset value from Code region or RAM region

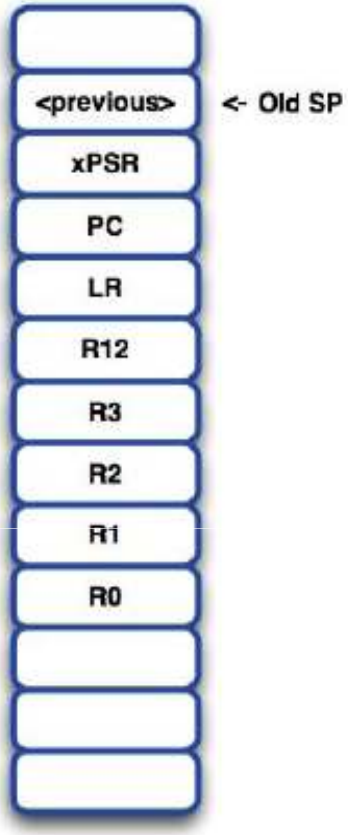
Az NVIC működése, az IT hatása I.

- Az IT hatására a Cortex M3 IT kezelő állapotba megy és lementi a regiszter készletet a stack-re.
  - Ez mikrokódban történik nincs szükség hozzá programozói beavatkozásra. 8 regiszter mentődik el: a
    - Program Status Register
    - Program Counter
    - Link Register
    - R0 – R3 regiszterek (ezek tartalmazzák a függvények paraméter hívásait) és az R12 regiszter (compiler segéd adatregiszter)
    - Process Stack átkapcsolódik a Main Stackre ha szükséges
  - Eközben az IT kiszolgáló címét elkezdi felhozni az utasítás buszon.

Az NVIC működése, az IT hatása II.

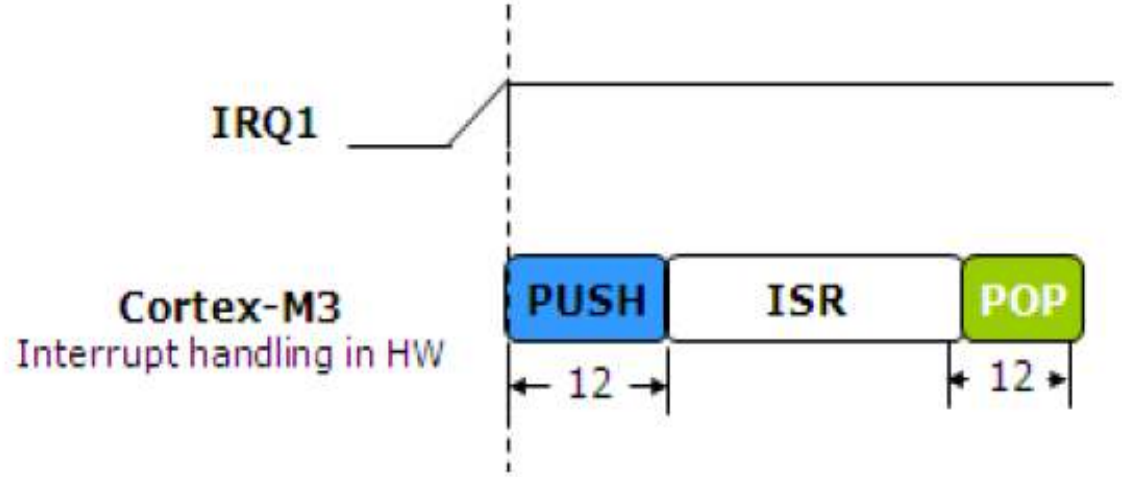
- Az IT kezdőcímének felhozása után aktualizálás
  - Stack Pointer
  - Link Register
  - Program Counter
  - Interupt Program Status Register: IPSR
- Az IT kiváltása után 12 órajellel elkezdődik az IT kiszolgálás.
- Az IT után a visszatérés ugyanúgy 12 órajel ciklus.
  - Nincs speciális visszatérő utasítás

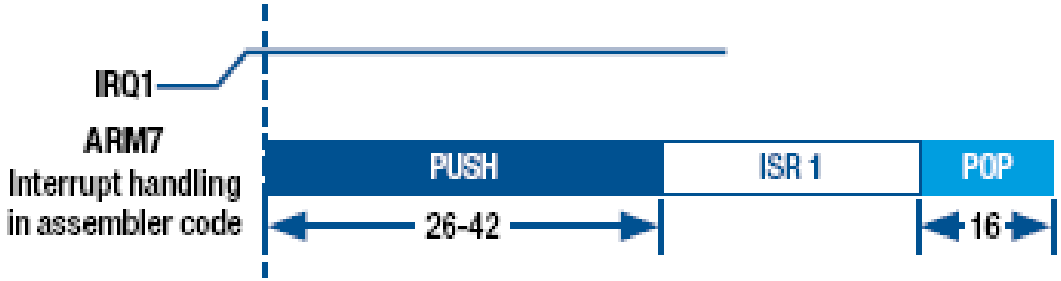
Az NVIC működése összefoglalás



<- Old SP

The NVIC will respond to an interrupt with a latency of just six cycles. This includes a microcoded routine to automatically push a set of registers onto the stack.



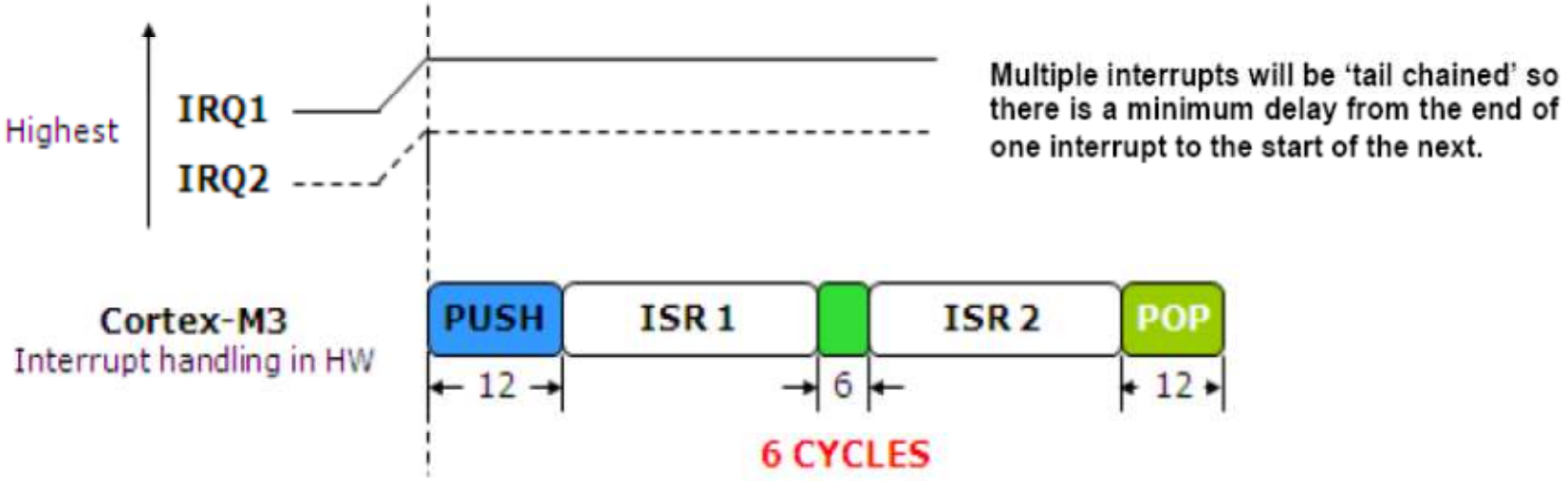


Működés több egyidejű megszakítás esetén I.

- Real-time alkalmazásokban fontos, hogy a megszakítások prioritását is szabályozni tudjuk.
  - Hagyományosan a hard-realtime rendszerek nem, vagy csak nagyon kevés IT-t használnak, pont azért mert a sok IT egymást is késlelteti.
- Preemptív IT kezelés
  - A Cortex M3 biztosítja, hogy egy magasabb prioritású IT meg tudjon szakítani egy alacsonyabb prioritásút.
  - Az alacsonyabb prioritású IT regiszterei, ugyanúgy mint a főprogram regiszterei, lementődnek. A magas prioritású megszakítás 12 órajel ciklus alatt el kezd végrehajtódni.

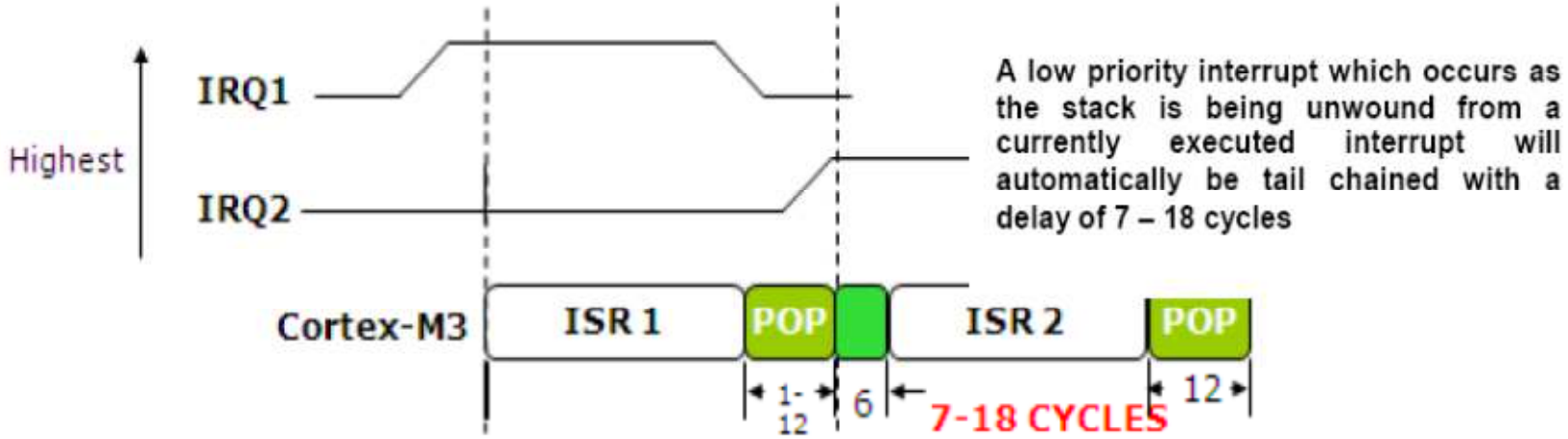
Működés több egyidejű megszakítás esetén II.

- Tail chaining: A megszakítások láncban történő végrehajtása több egyidejű IT esetén.
  - A második IT végrehajtása az ARM7 esetében jóval hosszabb lett volna (42 óraciklus), mert ott egy POP(16) és PUSH(26) is lejátszódott volna.



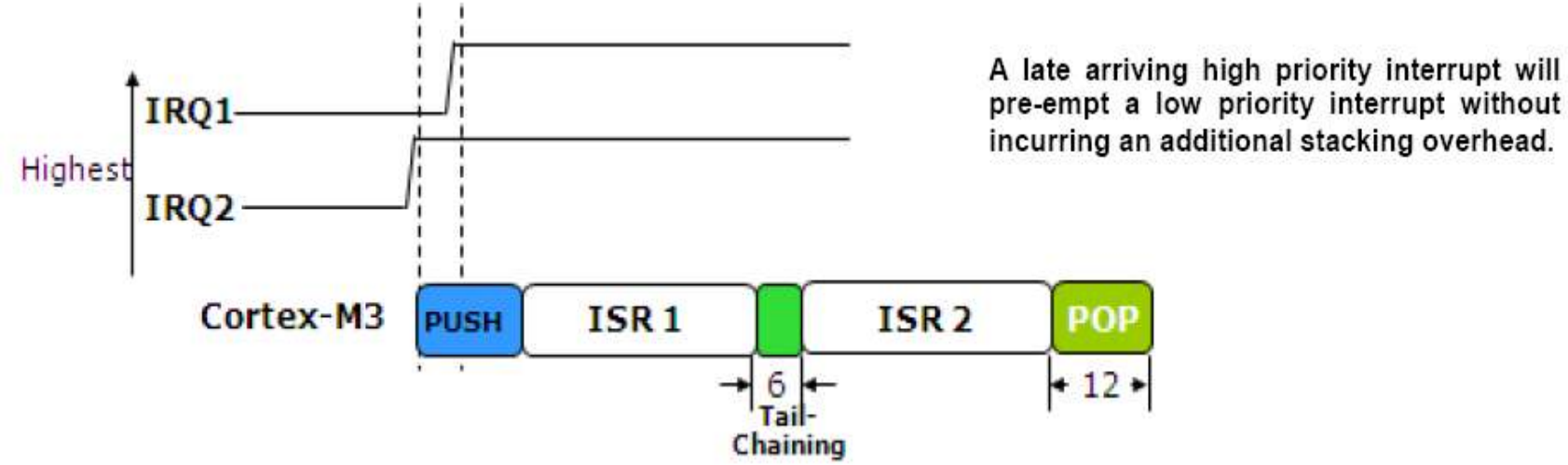
Működés több egyidejű megszakítás esetén III.

- Megszakításból való visszatérés megszakítása
  - A POP művelet itt megszakadhat.



Működés több egyidejű megszakítás esetén IV.

- Elkésett nagyobb prioritású IT nem okoz gondot.
  - A megszakítás kiváltása után 6 órajellel, ami az IT kiszolgáló címének felhozásához kell a nagyobb prioritású IT futáskész.



Nagyteljesítményű mikrovezérlők

8. DMA *(Direkt Memory Access)*

Scherer Balázs

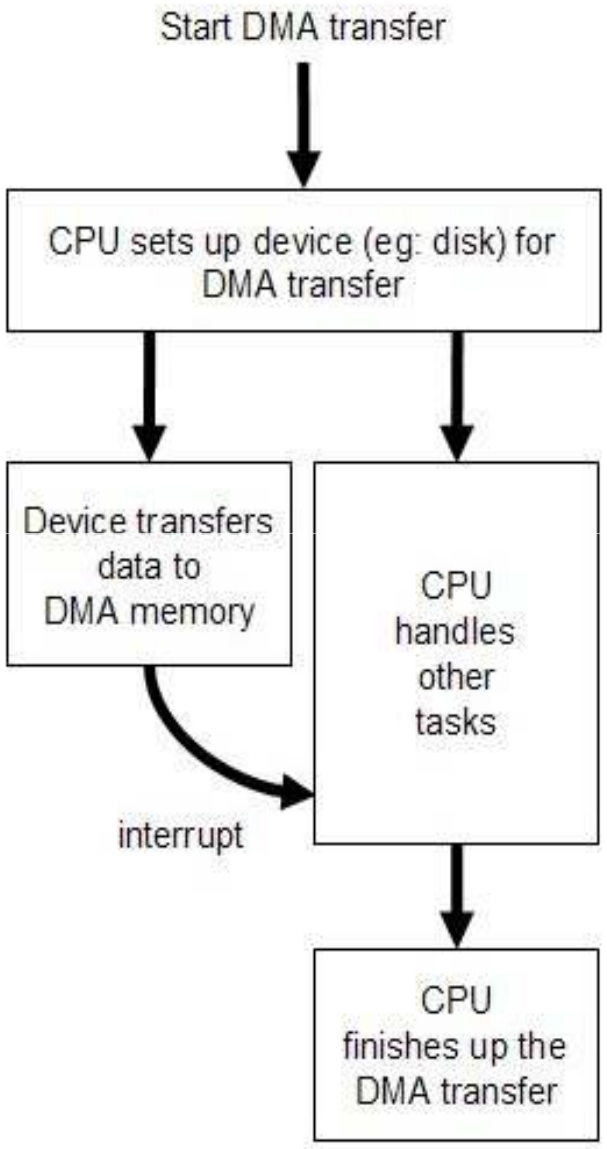
DMA áttekintés I.

Perifériák és memória blokkokok processzor beavatkozása nélkül hozzáférnek a rendszerbuszhoz.

- Periféria – Periféria
- Memória – Periféria
- Memória – Memória

- A DMA alkalmazása csökkentheti a processzor által kiszolgálandó megszakítások számát, továbbá magát a hardware-t is picit ésszerűsítheti, mert nem kell minden perifériához önálló FIFO-t rendelni.

DMA áttekintés II.



DMA áttekintés III.

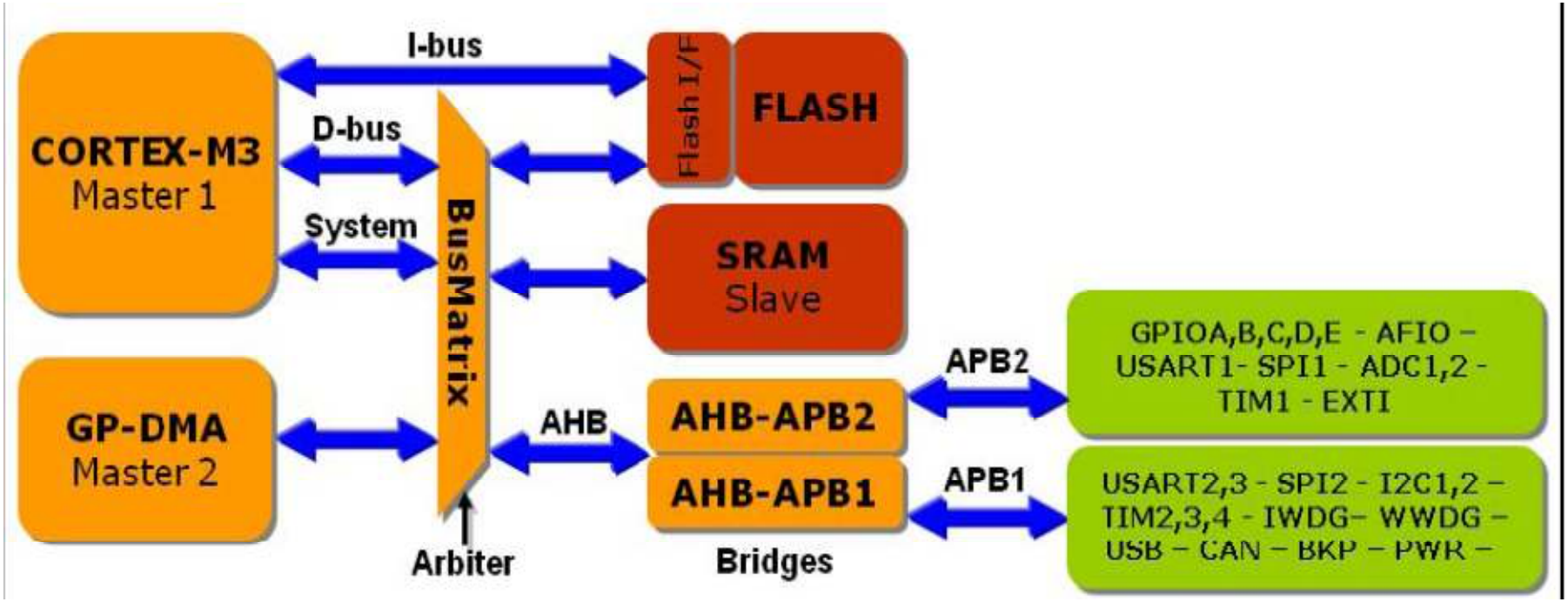
- Minden DMA ciklus tipikusan legalább két busz ciklust igényel
  - periféria olvasást
  - memória írást
- A DMA vezérlő semmilyen feldolgozást nem végez az adatokon.

DMA vezérlő programozása

- Szoftverből programozhatóak fel.
  - Átvitel forrás báziscíme
  - Cél báziscíme
  - Átviendő blokk hosszának beállítása
  - Ciklus végéhez tartozó megszakítás jelzés
  - Burst-ös, vagy egyciklusú hozzáférés
    - Burst: hatékonyabb de lassabb reakció külső eseményekre
  - A legtöbb DMA kontroller megvalósítás képes arra, hogy a cél és vagy a forrás címeket automatikusan inkrementálja.

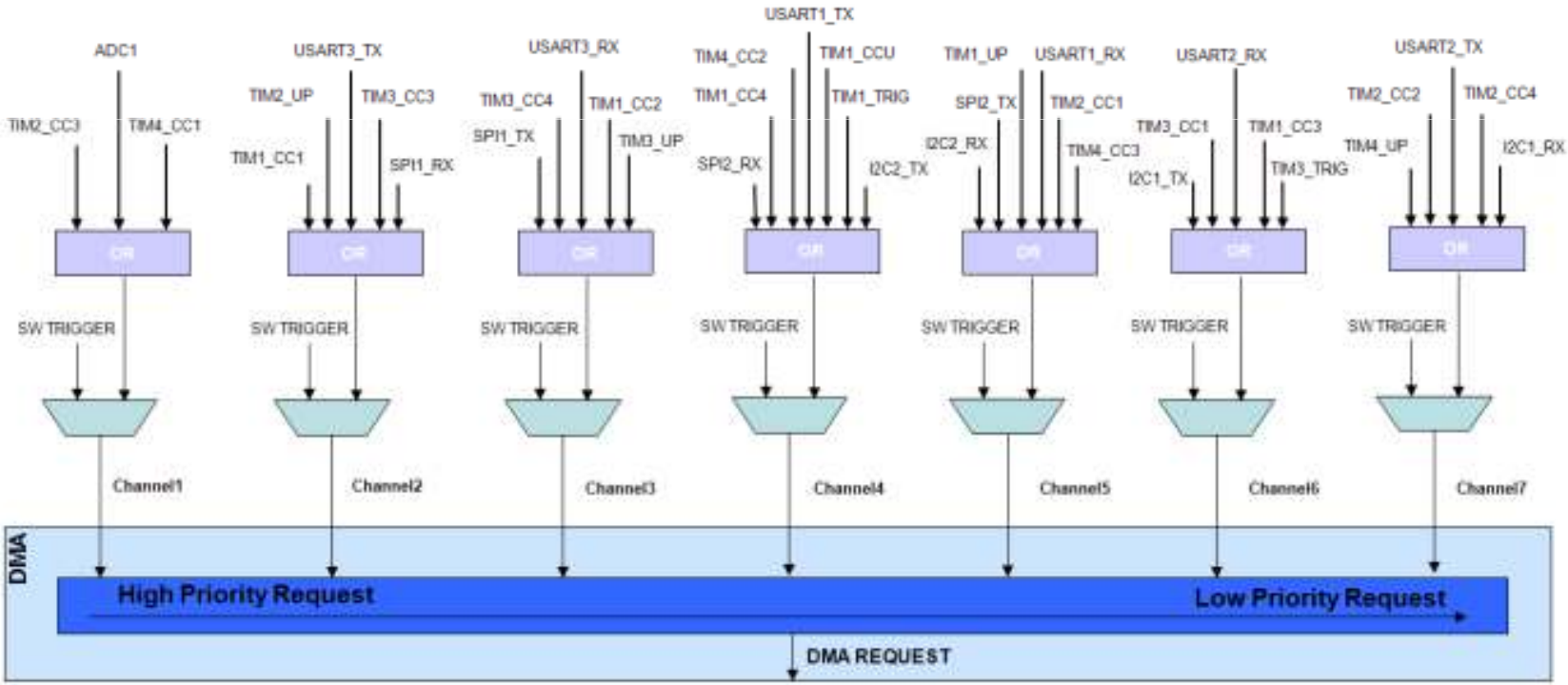
DMA az STM32 első generációjánál

- 1 DMA vezérlő
  - 12 csatorna
  - Csatornák dedikált eseményeket kezelnek
- Prioritás: *very high, high, medium, low*



Perifériák hozzárendelése a hardware csatornákhoz

- Előre el van döntve, hogy melyik periféria melyik hardware csatornán kérhet DMA átvitelt
- Egy csatornához több periféria is hozzá van rendelve így vigyázni kell, hogy milyen átviteli konfigurációk lehetségesek egyáltalán



DMA csatornakonfigurálásának folyamata

- A periféria regiszter címének beállítása a DMA_CPARx regiszterben. Az adat innen, vagy ide fog íródni a periféria esemény hatására.
- A memória cím beállítása a DMA_CMARx regiszterben. Az adat ide, vagy innen fog íródni a periféria esemény hatására.
- Az átviendő adatok számának konfigurálása a DMA_CNDTRx regiszterben. Minden periféria esemény hatására ez dekrementálódni fog.
- A csatorna priorításának meghatározása a DMA_CCRx regiszterben: *very high, high, medium, low.*
- Az adatátvitel irányának konfigurálása, a periféria és memória adatméret és inkrement konfigurálása, a cirkuláris mód konfigurálása, ha szükséges és a megszakítási események konfigurálása (átvitel felénél, az átvitel végén stb.).
- A csatorna aktivitálása a DMA_CCRx-regiszter enable bitjével.

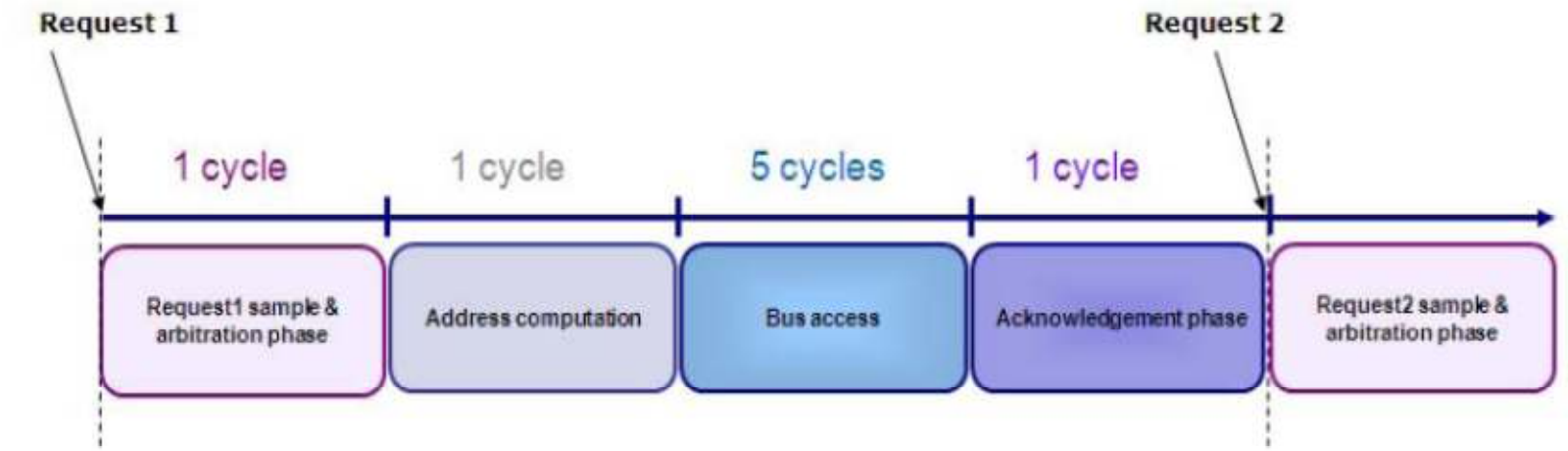
DMA átvitel folyamata STM32

Egy általános DMA átvitel a következőképpen zajlik:

- Az adat betöltése a belső címregiszter segítségével
  - (DMA_CPARx vagy DMA_CMARx) a periféria adatregiszteréből, vagy egy memória területről.
- A belső célcím regiszter segítségével az adat letárolódik a megadott memória, vagy periféria címre.
- A szükséges DMA átviteli ciklusok számának csökkentése
  - A cél és a forrás címek opcionális inkrementálása (az inkrementálás a konfigurált adatmérettől függően 1,2, vagy 4 byte-al inkrementálódhat).
- Ha az utolsó ciklus is végrehajtódott, és nem cirkuláris módban volt konfigurálva a DMA, akkor a következő adatátvitel előtt ki kell kapcsolni az adott csatornát.
  - Cirkuláris módban az utolsó adattranszfer után a cél és a forrás címek visszaállnak a kezdeti értékre.

DMA ciklusok időigénye I.

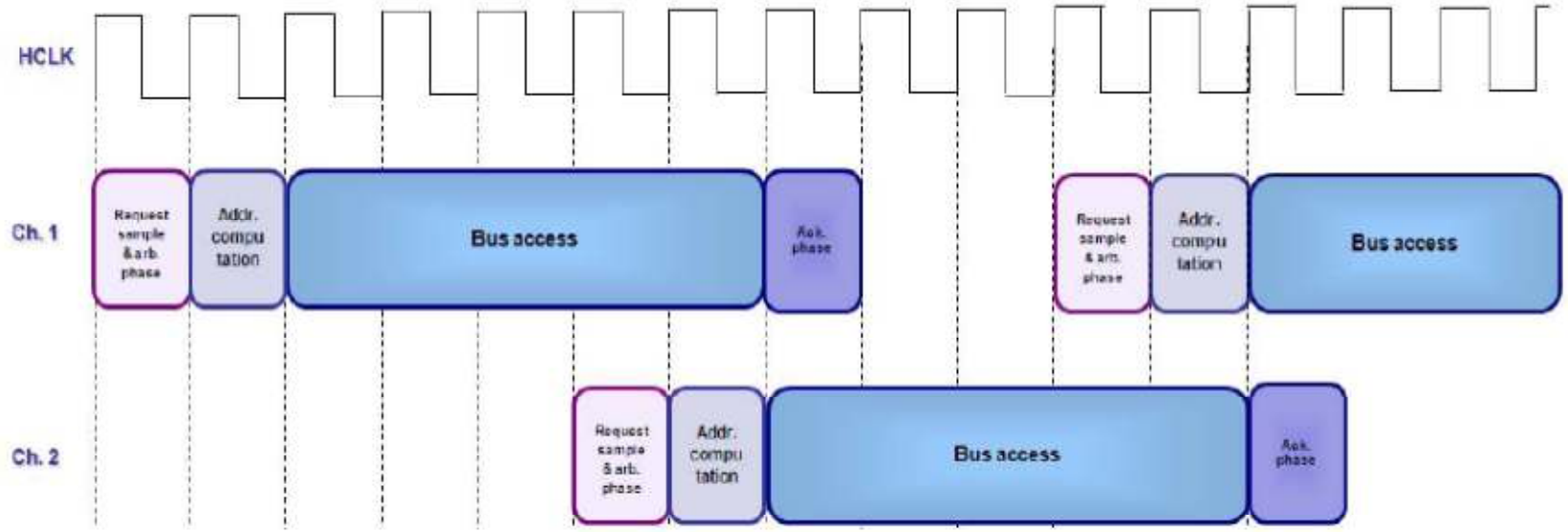
- Az STM32 DMA-ja kis méretű blokkok átvitelére optimalizált (memória - memória).



Each DMA memory to memory transfer is made up of four phases each 1 cycle long except for the bus access phase that takes five cycles for each word transferred

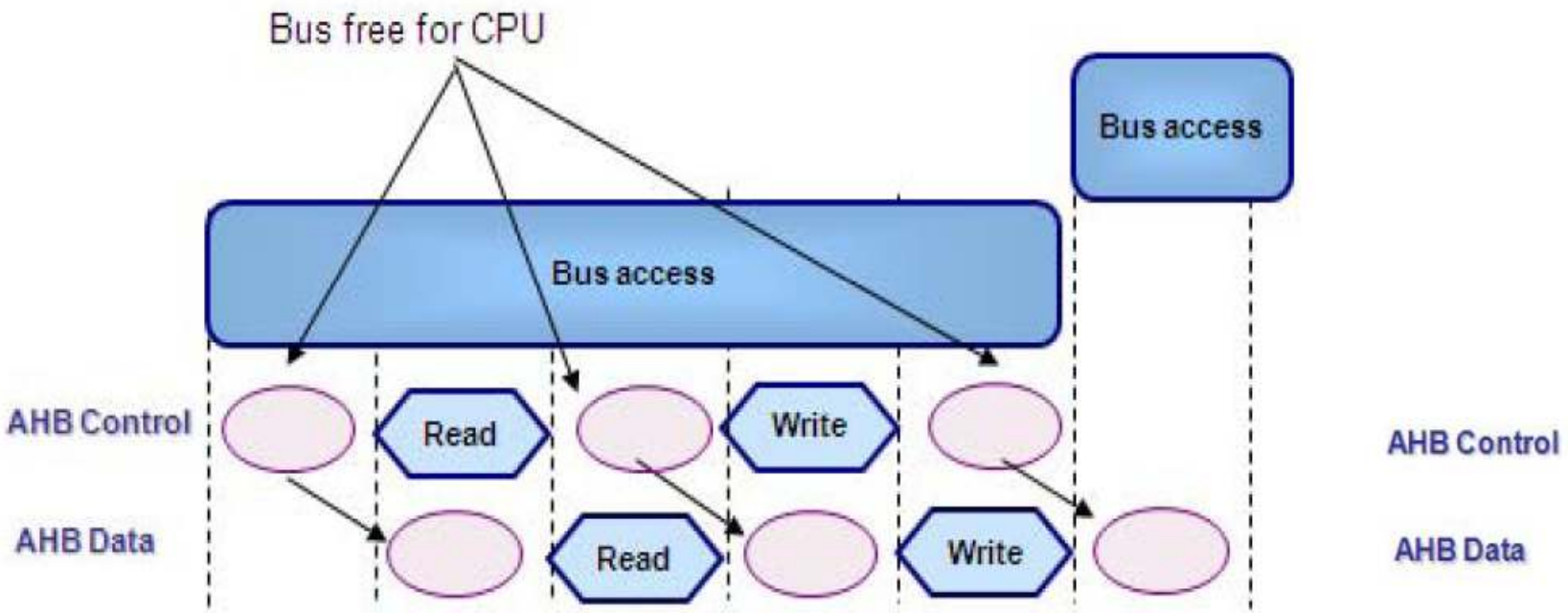
DMA ciklusok időigénye II.

- A cím kiszámolási és nyugtázási fázis történhet párhuzamosan a buszon lévő kommunikációval.



DMA ciklusok időigénye III.

- A Bus Acces fázis 5 ütemig tart, mert a CPU-val együttműködve használja a buszt (memória - memória)



During the bus access phase three cycles per transfer are free for the CPU. In a memory to memory transfer this guarantees the Cortex-M3 60% of the bus even when the DMA is running continuously (remember this is for data transfer only). The Cortex has a separate I-code bus to fetch instructions.

Periféria – Memória

- AHB feletti transzfer ciklus 2 órajelciklust igényel az AHB frekvenciáján
- Az APB transzfer ciklus szintén 2 órajelciklust igényelnek, de az APB frekvenciáján, és még kettőt az AHB frekvenciáján.
- SPI – memória DMA ciklus:

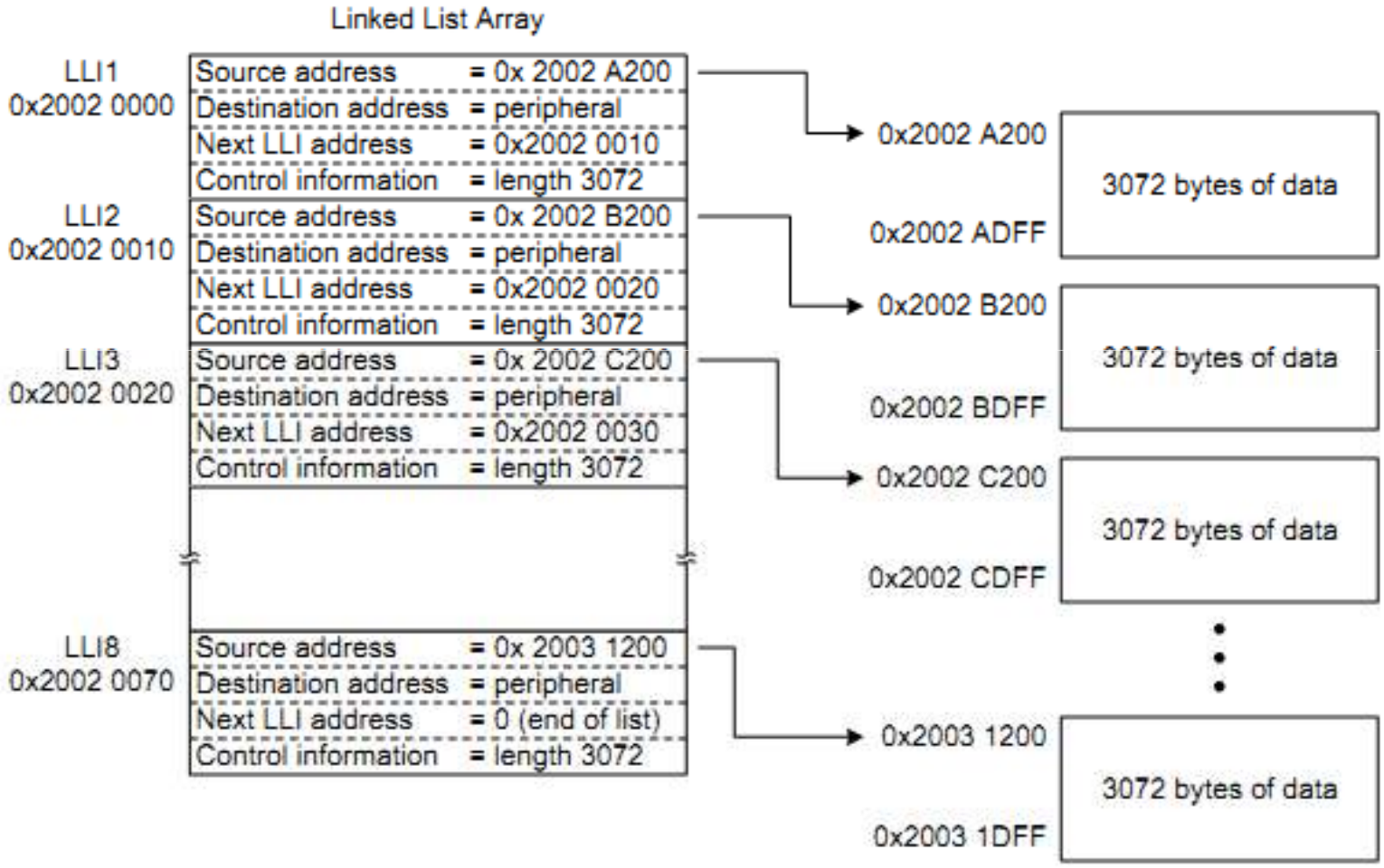
SPI to SRAM DMA transfer = SPI transfer (APB) + SRAM transfer (AHB) + free cycle(AHB)
= (2 APB cycles + 2 AHB cycles) + 2 AHB cycles + 1 AHB cycle
= 2APB Cycles + 5 AHB cycles

LPC 1768

- Ugyanaz, mint az STM32
  - 16 forrás vonal
  - Scatter and Gather DMA (nem szükséges folytatólagos buffer linkelt lista is lehet)
  - DMA működhet bizonyos energiatakarékos módokban

LPC 1768

- Scatter and gather DMA működés



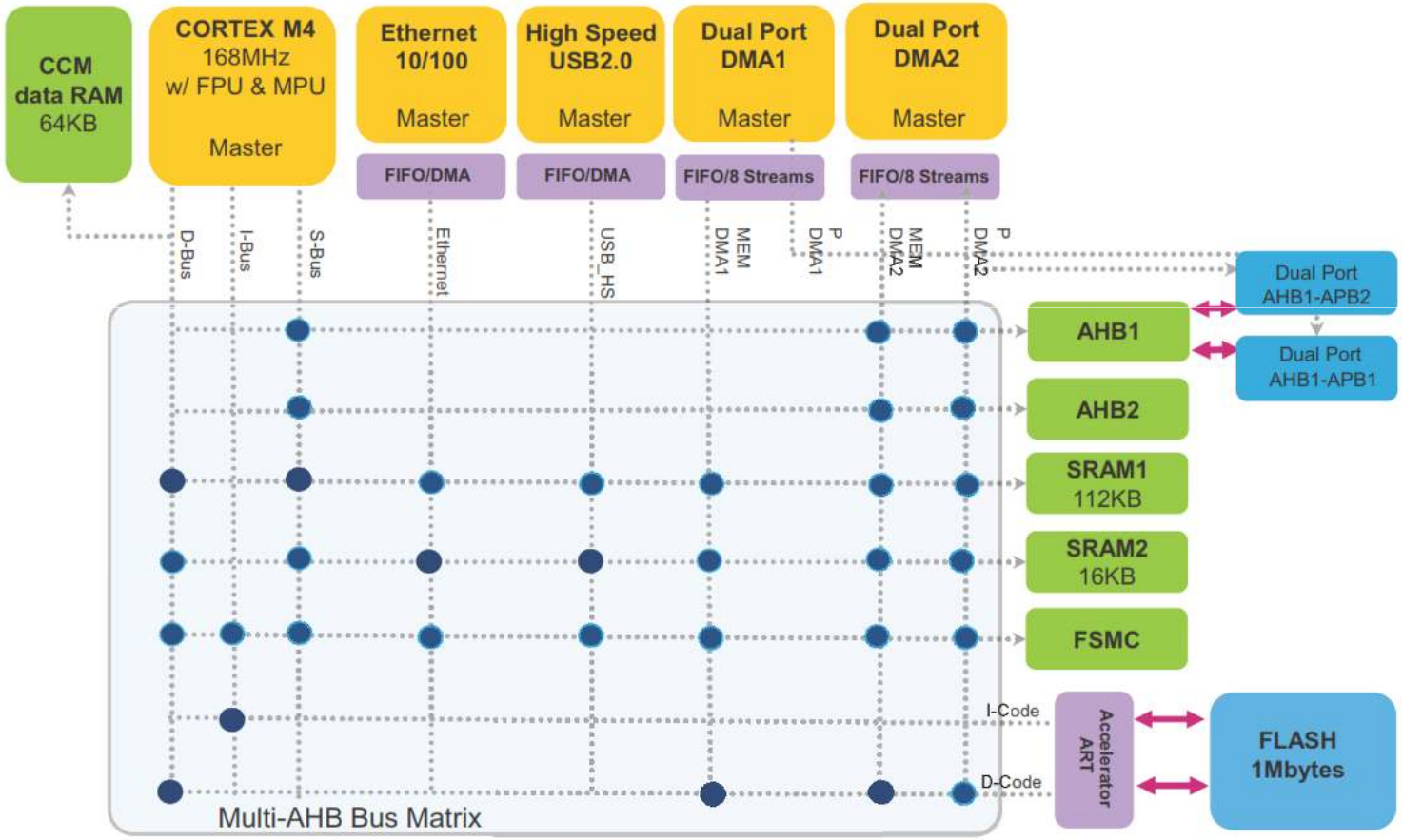
LPC DMA program vs STM

```
/* preemption = 1, sub-priority = 1 */
NVIC_SetPriority(DMA_IRQn, ((0x01<<3)|0x01));

/* Initialize GPDMA controller */
    GPDMA_Init();

    // Setup GPDMA channel ---------------------
    // channel 0
    GPDMACfg.ChannelNum = 0;
    // Source memory
    GPDMACfg.SrcMemAddr = DMA_SRC;
    // Destination memory
    GPDMACfg.DstMemAddr = DMA_DST;
    // Transfer size
    GPDMACfg.TransferSize = DMA_SIZE;
    // Transfer width
    GPDMACfg.TransferWidth = GPDMA_WIDTH_WORD;
    // Transfer type
    GPDMACfg.TransferType = GPDMA_TRANSFERTYPE_M2M
    // Source connection - unused
    GPDMACfg.SrcConn = 0;
    // Destination connection - unused
    GPDMACfg.DstConn = 0;
    // Linker List Item - unused
    GPDMACfg.DMALLI = 0;
    // Setup channel with given parameter
    GPDMA_Setup(&GPDMACfg);

    /* Reset terminal counter */
    Channel0_TC = 0;
    /* Reset Error counter */
    Channel0_Err = 0;

    _DBG_("Start transfer...");

    // Enable GPDMA channel 0
    GPDMA_ChannelCmd(0, ENABLE);
```

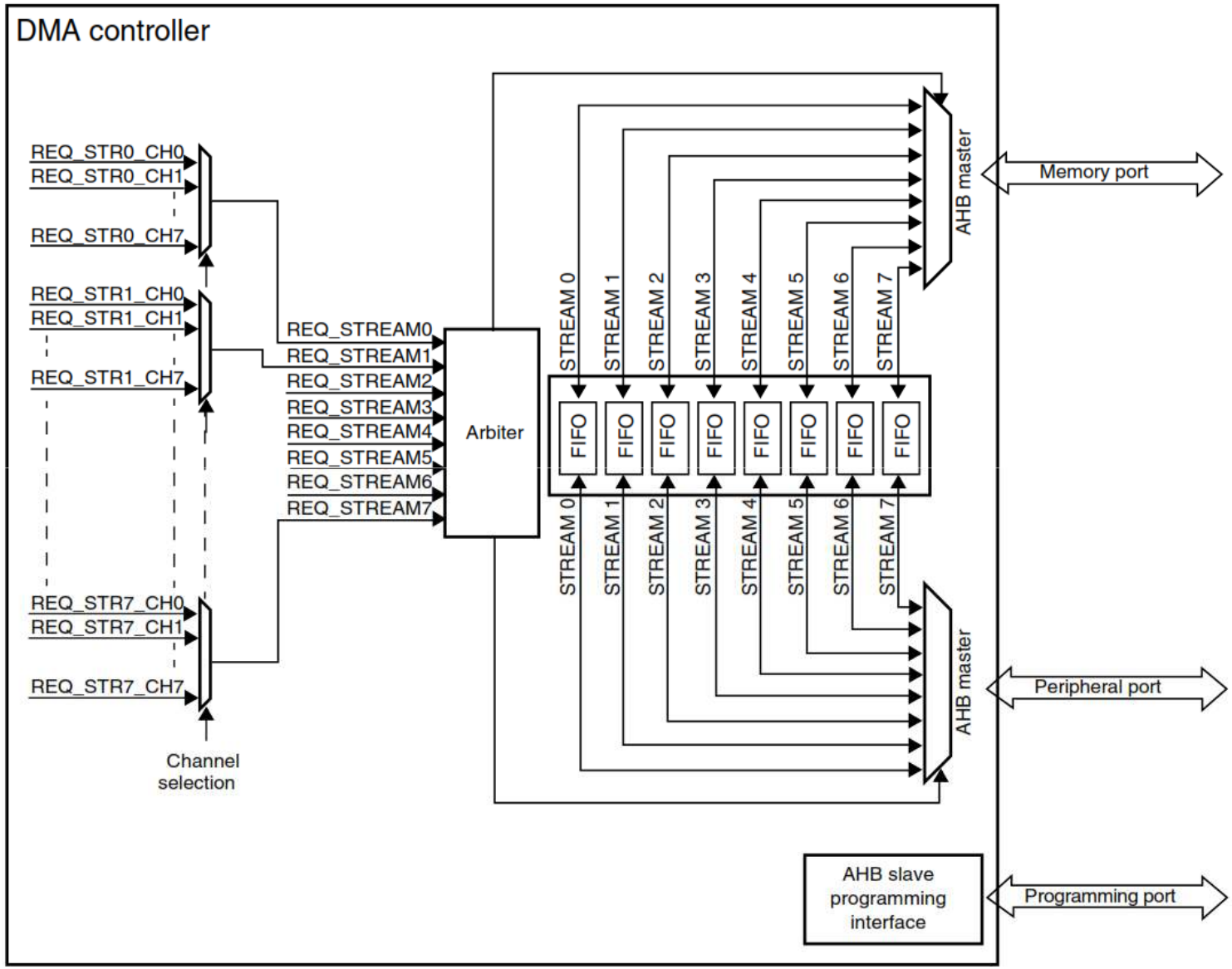
```
int main(void)
{
  /* System Clocks Configuration */
  RCC_Configuration();

  /* NVIC configuration */
  NVIC_Configuration();

  /* DMA1 channel6 configuration */
  DMA_DeInit(DMA1_Channel6);
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)SRC_Const_Buffer;
  DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)DST_Buffer;
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  DMA_InitStructure.DMA_BufferSize = BufferSize;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Enable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
  DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
  DMA_InitStructure.DMA_Priority = DMA_Priority_High;
  DMA_InitStructure.DMA_M2M = DMA_M2M_Enable;
  DMA_Init(DMA1_Channel6, &DMA_InitStructure);

  /* Enable DMA1 Channel6 Transfer Complete interrupt */
  DMA_ITConfig(DMA1_Channel6, DMA_IT_TC, ENABLE);

  /* Get Current Data Counter value before transfer begins */
  CurrDataCounterBegin = DMA_GetCurrDataCounter(DMA1_Channel6);

  /* Enable DMA1 Channel6 transfer */
  DMA_Cmd(DMA1_Channel6, ENABLE);

  /* Wait the end of transmission */
  while (CurrDataCounterEnd != 0)
  {
  }
```

DMA az STM32F2, STM32F4 generációnál

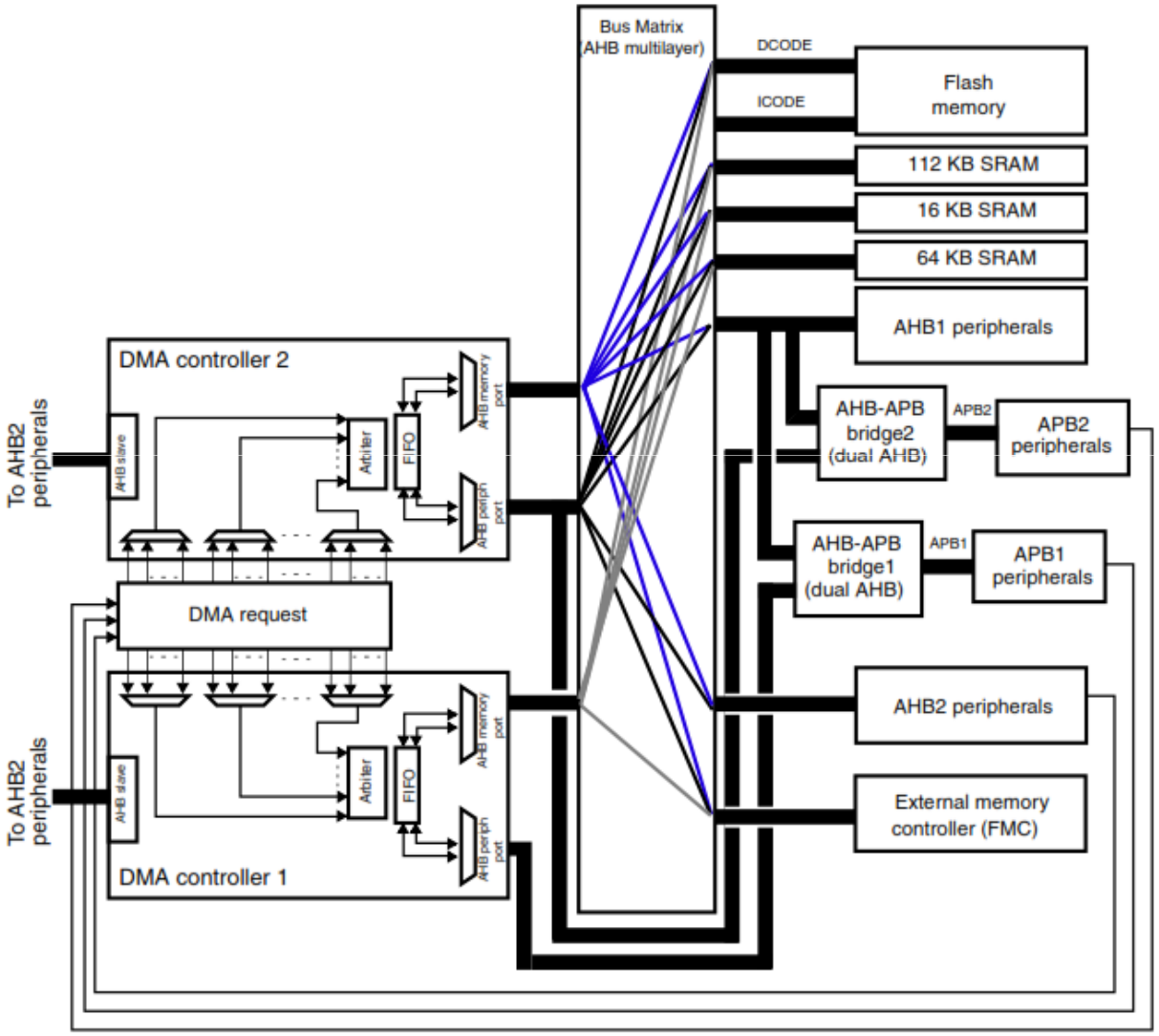
- 2 DMA vezérlő dedikált hozzáférésekkel is, más masterek is tudnak DMA-zni



A DMA vezérlők belső felépítése



A DMA vezérlők belső felépítése



Csatornák szintén eszköz specifikusak

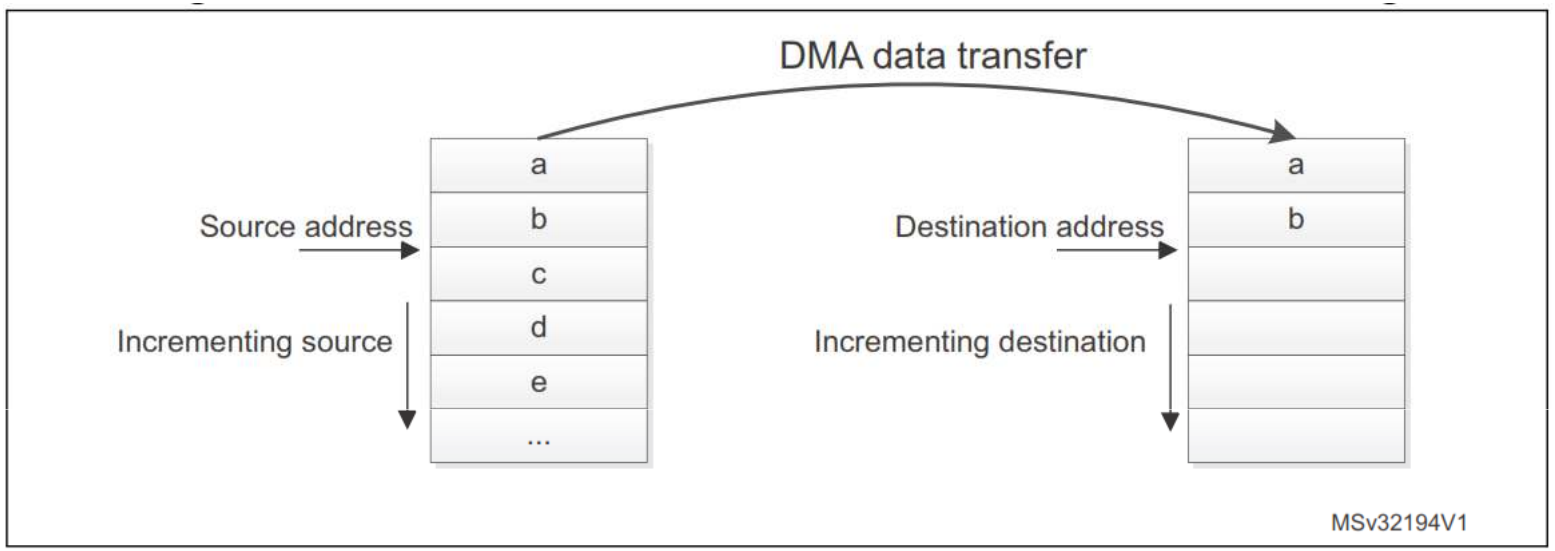
- Vigyázni, hogy ki kivel tud szóbaállni

Table 2. DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	-	TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

DMA tulajdonságok

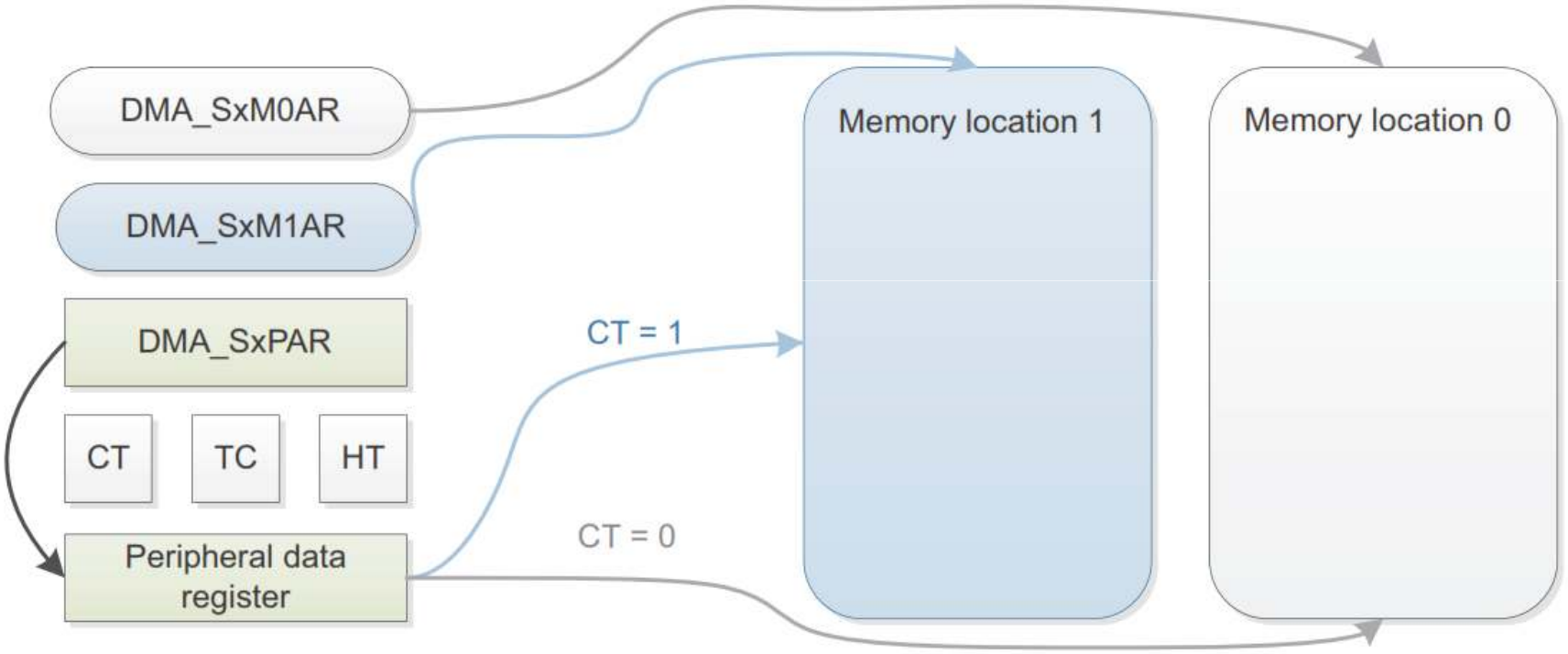
- Cím inkrementálás



- Adathosszúság kezelés
  - Byte (8 bit)
  - Half-Word (16bit)
  - Word (32 bit)

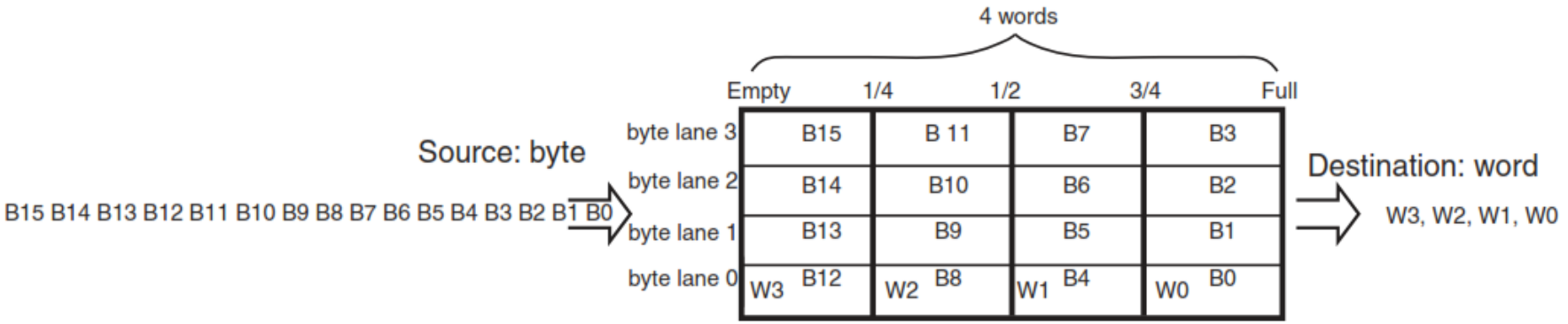
Dupla bufferelés

- Tipikus online feldolgozási feladatra optimalizálva



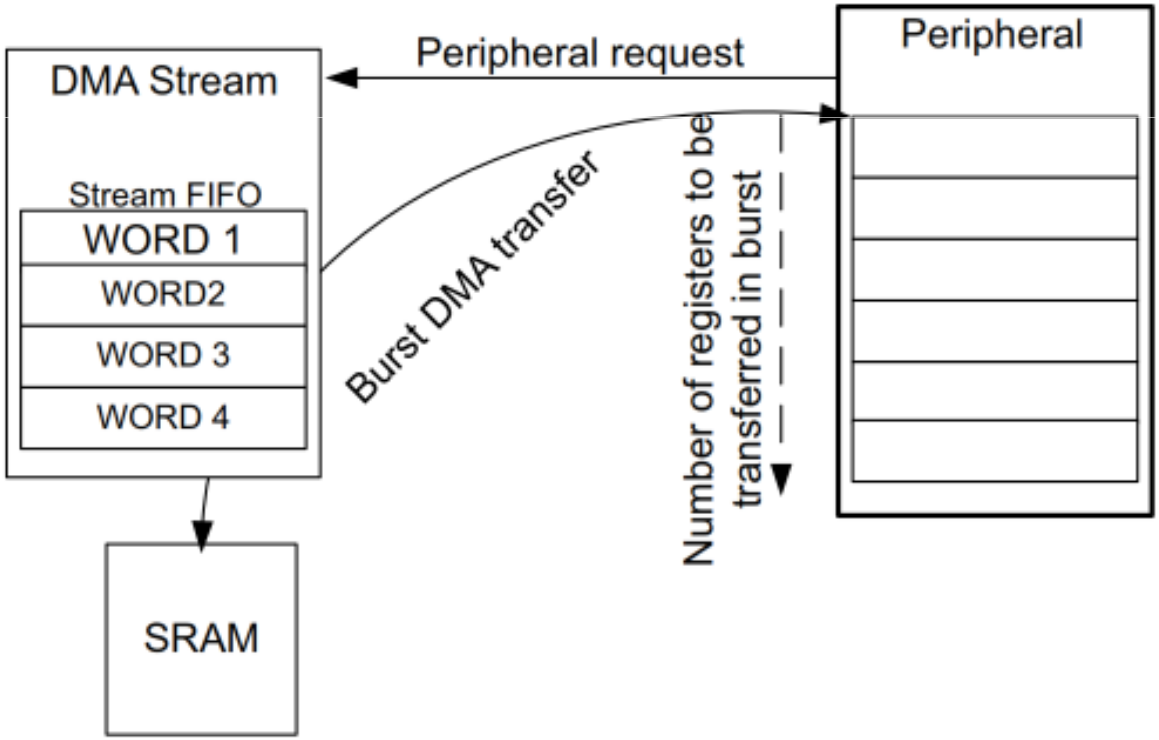
DMA FIFO

- 4 db 32 bites FIFO
  - Treshold konfigurálható ¼, ½, ¾ szintre
  - Packing – Unpacking lehetőség
  - Burst-ös átvitel lehetőség



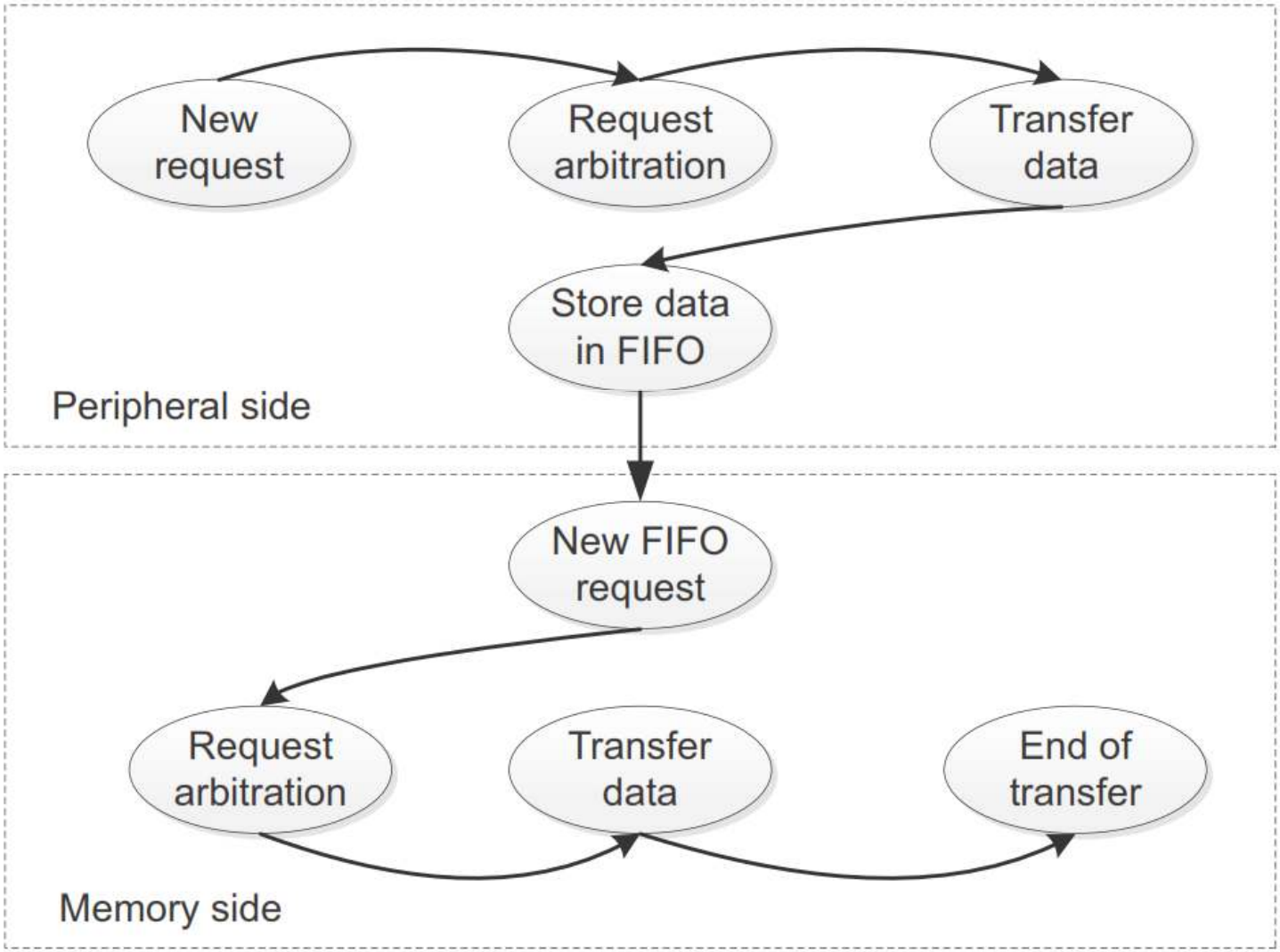
DMA burst

- 4 db 32 bites FIFO méretéig konfigurálható max.
  - 16 byte, 8 half-word, 4 word max.
  - Lehet 4 increment (4 byte, 4 half word, 4 word), 8 incremen (8 byte, 8 half-word)t, 16-os increment (8 byte)



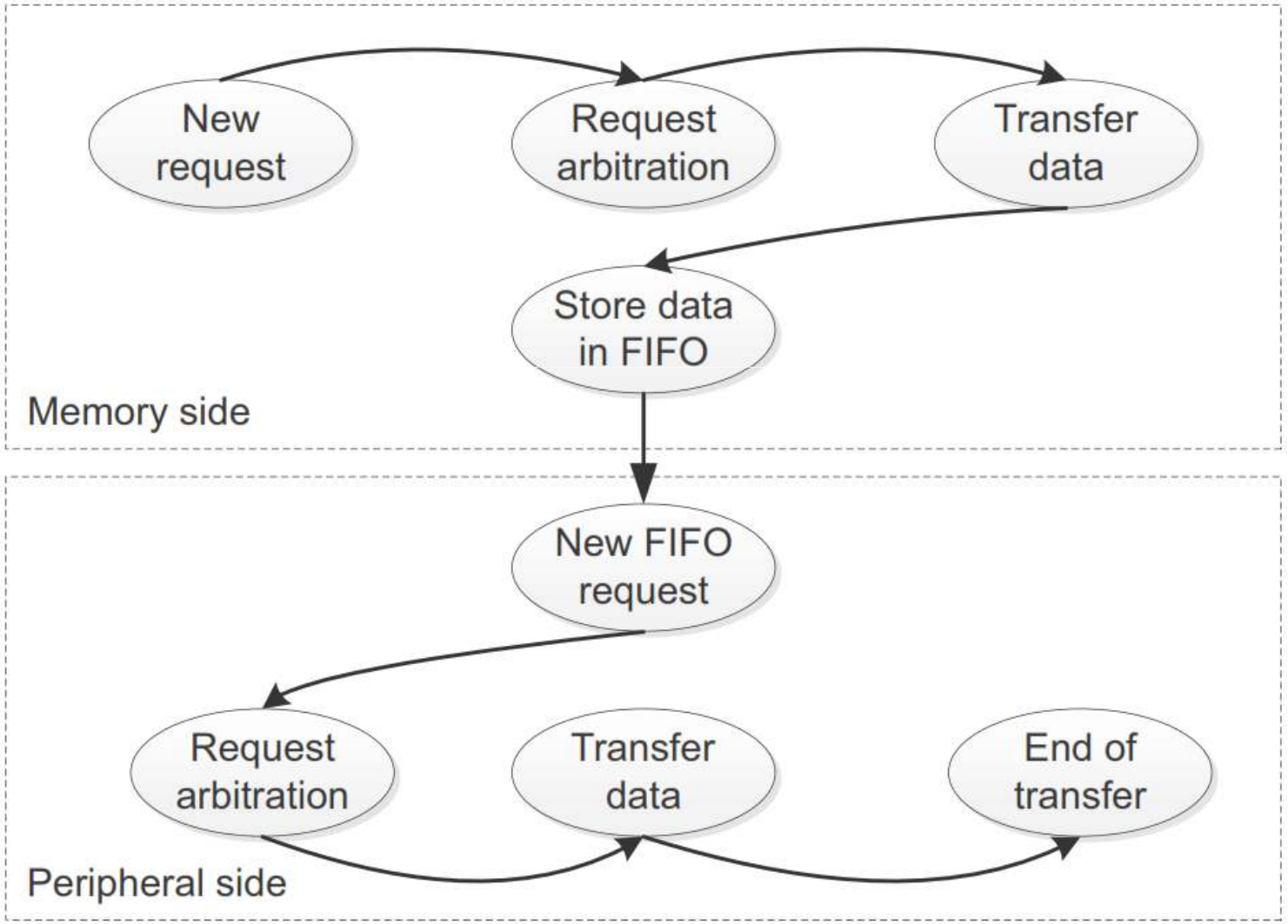
DMA átviteli állapotok

- Periféria – Memória út: 2 buszhozzáférés



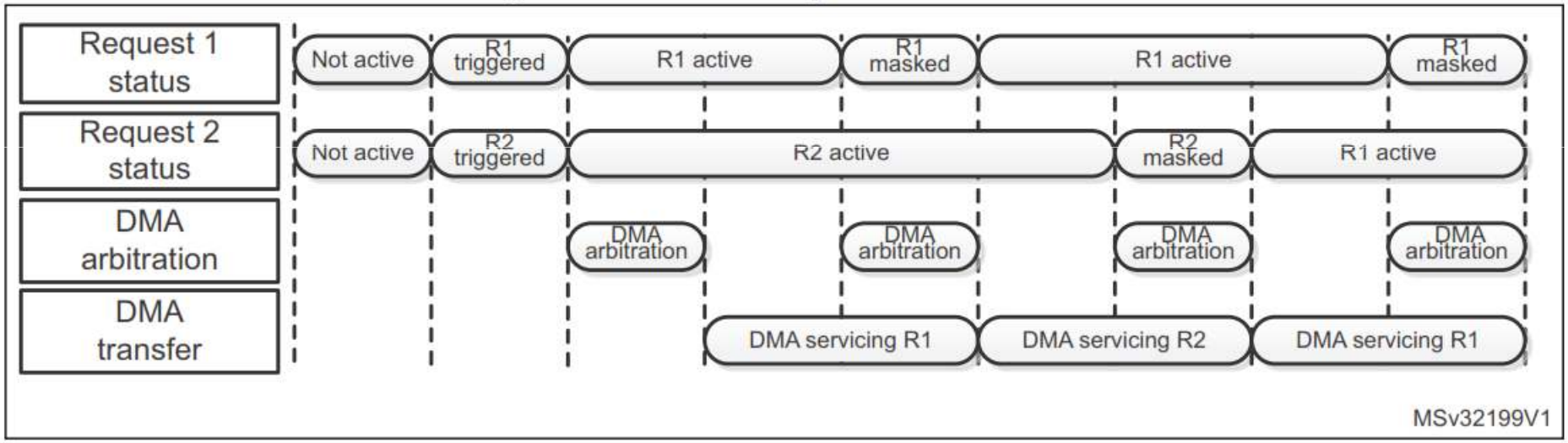
DMA átviteli állapotok

- Periféria – Memória út: 2 buszhozzáférés



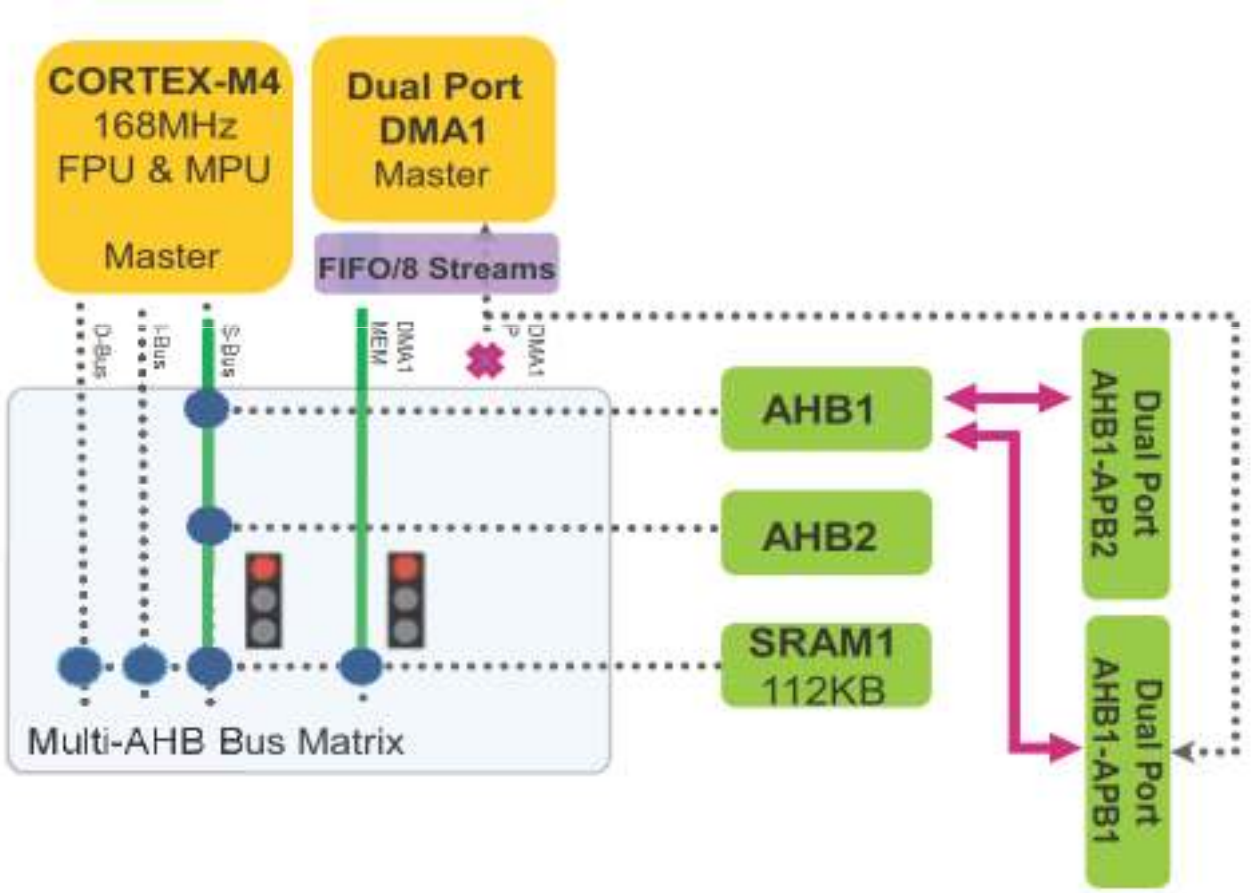
DMA kérések arbitrációja

- Periféria – Memória út: 2 buszhozzáférés
  - Request 1 a magasabb priorítású



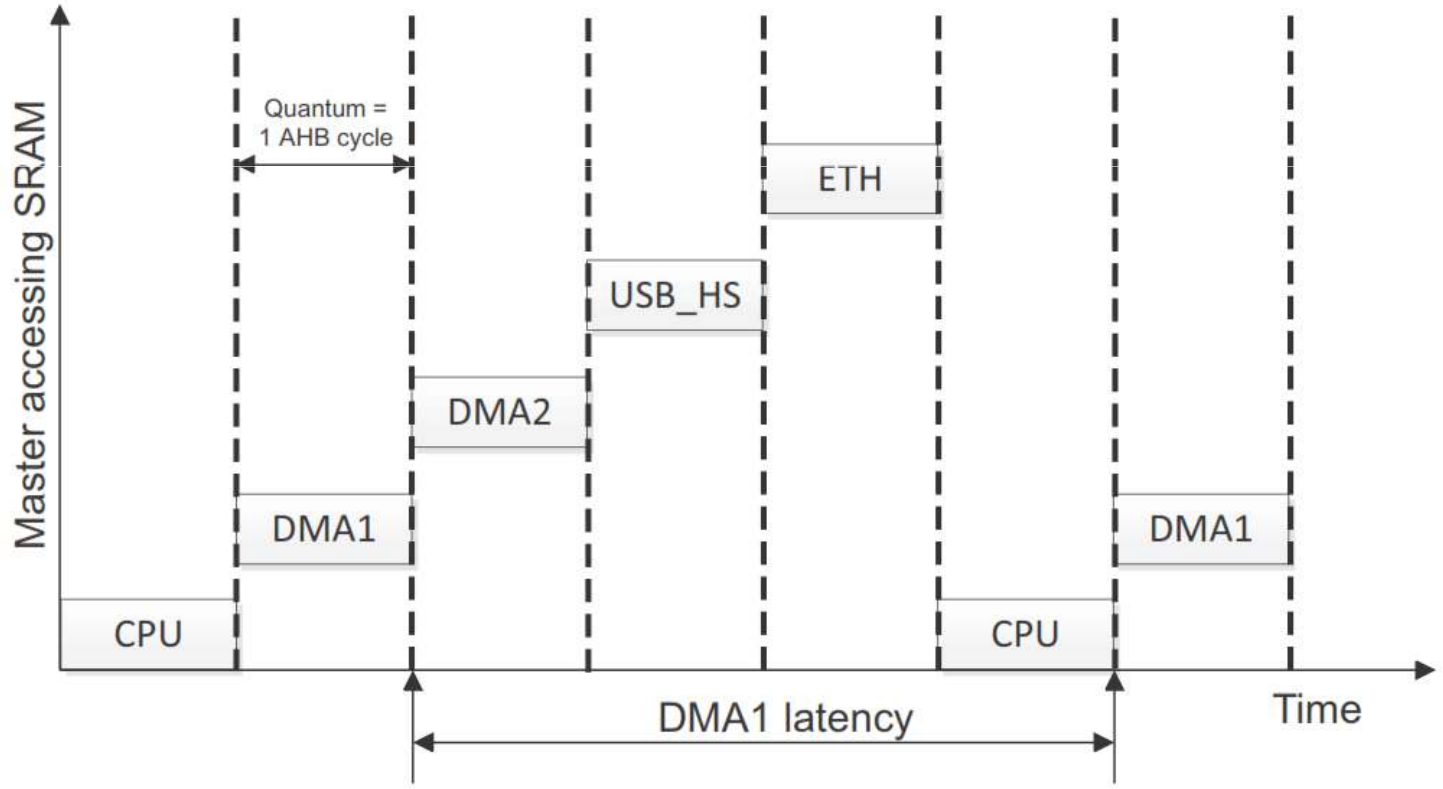
Buszmátrix hozzáférés

- Ütközés esetén round-robin hozzáférés
  - SRAM hozzáférési minta



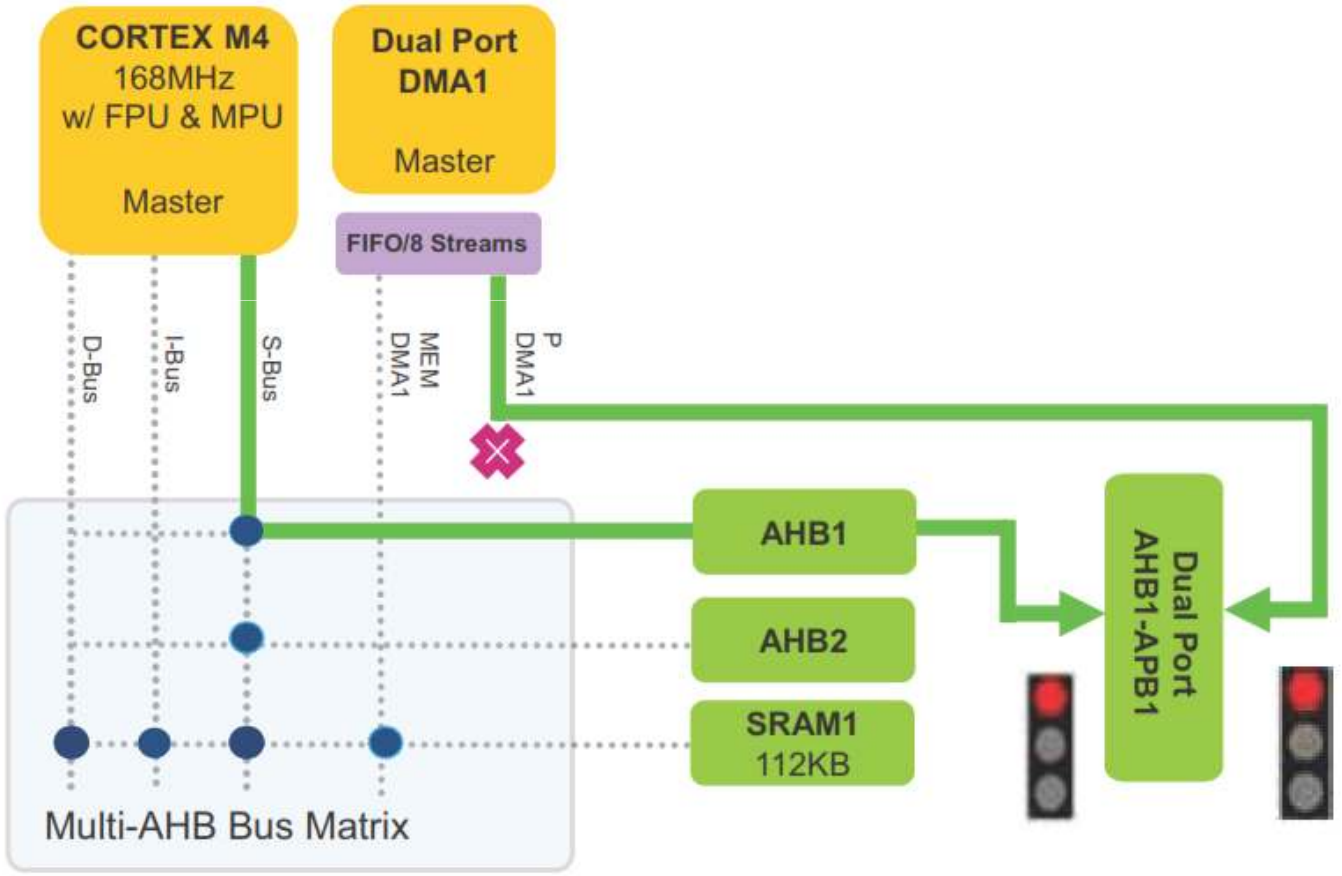
Buszmátrix ütközések ideje

- Mindig az adott ágon várakozók számától függ
  - A CPU több ciklusig is foglalhatja a buszt
    - Interrupt: 8 AHB ciklus
    - Multiple Store, Load akár 14 AHB ciklus is



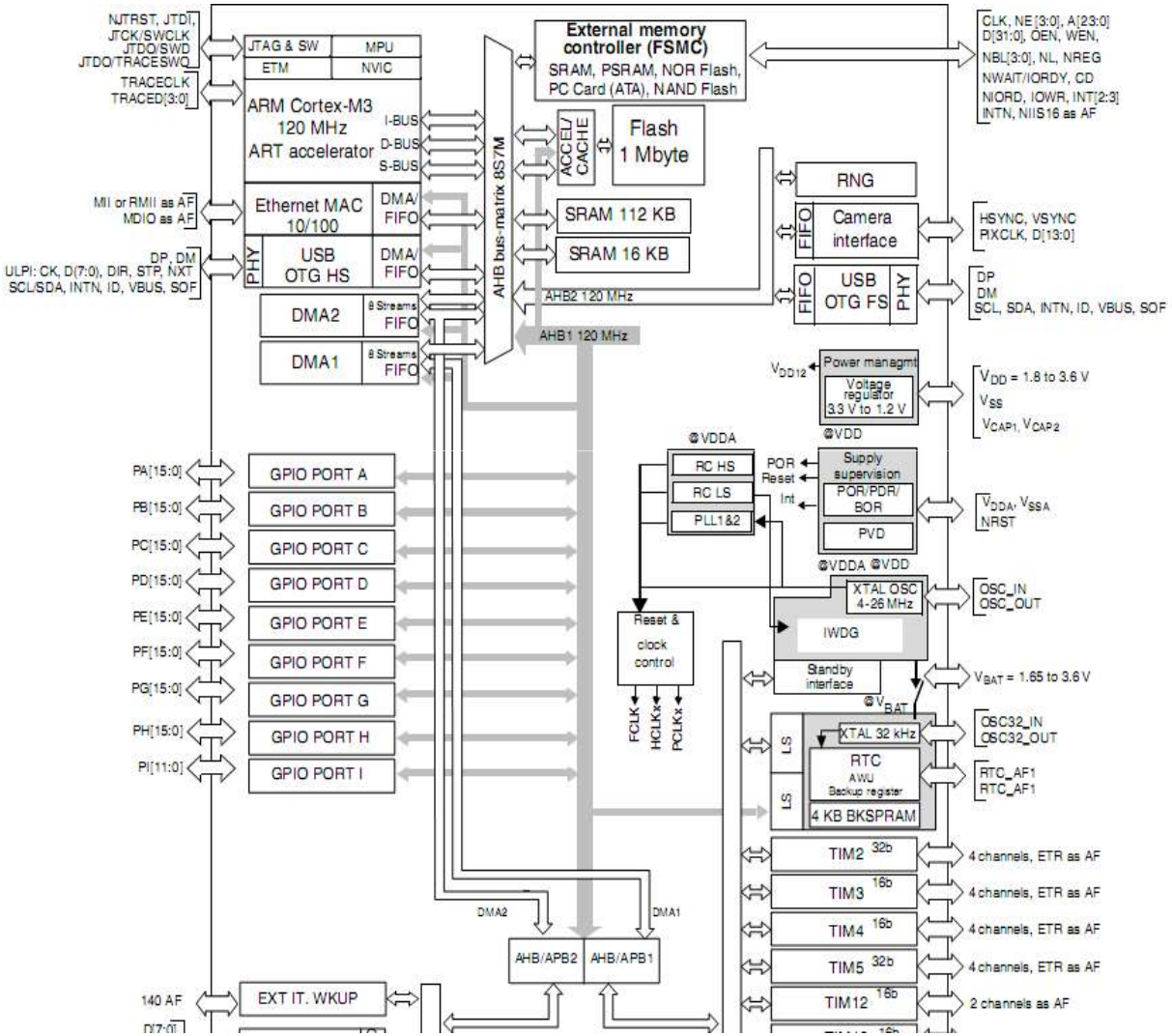
Dual portos AHB-APB bridgek

- Az APB port-hoz csak egyszerre csak egy Master férhet hozzá



Mi értelme van a Dual port APB bridge-nek?

- 2010: Az STM32F2xx belső architektúrája



DMA időzítések

- Hozzáférési idő

$T_S = T_{SP}$ (peripheral access/transfer time) + T_{SM} (memory access/transfer time)

- Periféria hozzáférés részletesebben

$T_{SP} = t_{PA} + t_{PAC} + t_{BMA} + t_{EDT} + t_{BS}$

Description	Through bus matrix		DMA's direct paths
	To AHB peripherals	To APB peripherals	
t_{PA}: DMA peripheral port arbitration	1 AHB cycle	1 AHB cycle	1 AHB cycle
t_{PAC}: peripheral address computation	1 AHB cycle	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration (when no concurrency)(1)	1 AHB cycle	1 AHB cycle	N/A
t_{EDT}: effective data transfer	1 AHB cycle(2) (3)	2 APB cycles	2 APB cycle
t_{BS}: bus synchronization	N/A	1 AHB cycle	1 AHB cycle

DMA időzítések

- Hozzáférési idő

$T_S = T_{SP}$ (peripheral access/transfer time) + T_{SM} (memory access/transfer time)

- Memória hozzáférés részletesebben

$T_{SM} = t_{MA} + t_{MAC} + t_{BMA} + t_{SRAM}$

Description	Latency
t_{MA}: DMA memory port arbitration	1 AHB cycle
t_{MAC}: memory address computation	1 AHB cycle
t_{BMA}: bus matrix arbitration (when no concurrency)(1)	1 AHB cycle(2)
t_{SRAM}: SRAM read or write access	1 AHB cycle

DMA időzítések példa

- ADC – SRAM átvitel
  - Periféria port késleltetés

AHB/APB2 frequency Transfer time	F_{AHB} = 72 MHz/ F_{APB2} = 72 MHz AHB/APB ratio = 1	FAHB = 144 MHz/ FAPB2 = 72 MHz AHB/APB ratio = 2
t_{PA}: DMA peripheral port arbitration	1 AHB cycle	1 AHB cycle
t_{PAC}: peripheral address computation	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration	N/A(1)	N/A(1)
t_{EDT}: effective data transfer	2 AHB cycles	4 AHB cycles
t_{BS}: bus synchronization	1 AHB cycle	1 AHB cycle
T_{SP}: total DMA transfer time for peripheral port	5 AHB cycles	7 AHB cycles

  - Memória port késleltetés

CPU/APB2 frequency Transfer time	FAHB = 72MHz/ FAPB2=72MHz AHB/APB ratio = 1	FAHB = 144MHz/ FAPB2=72MHz AHB/APB ratio = 2
t_{MA}: DMA memory port arbitration	1 AHB cycle	1 AHB cycle
t_{MAC}: memory address computation	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration	1 AHB cycle(1)	1 AHB cycle(1)
t_{SRAM}: SRAM write access	1 AHB cycle	1 AHB cycle
T_{SM}: total DMA transfer time for memory port	4 AHB cycles	4 AHB cycles