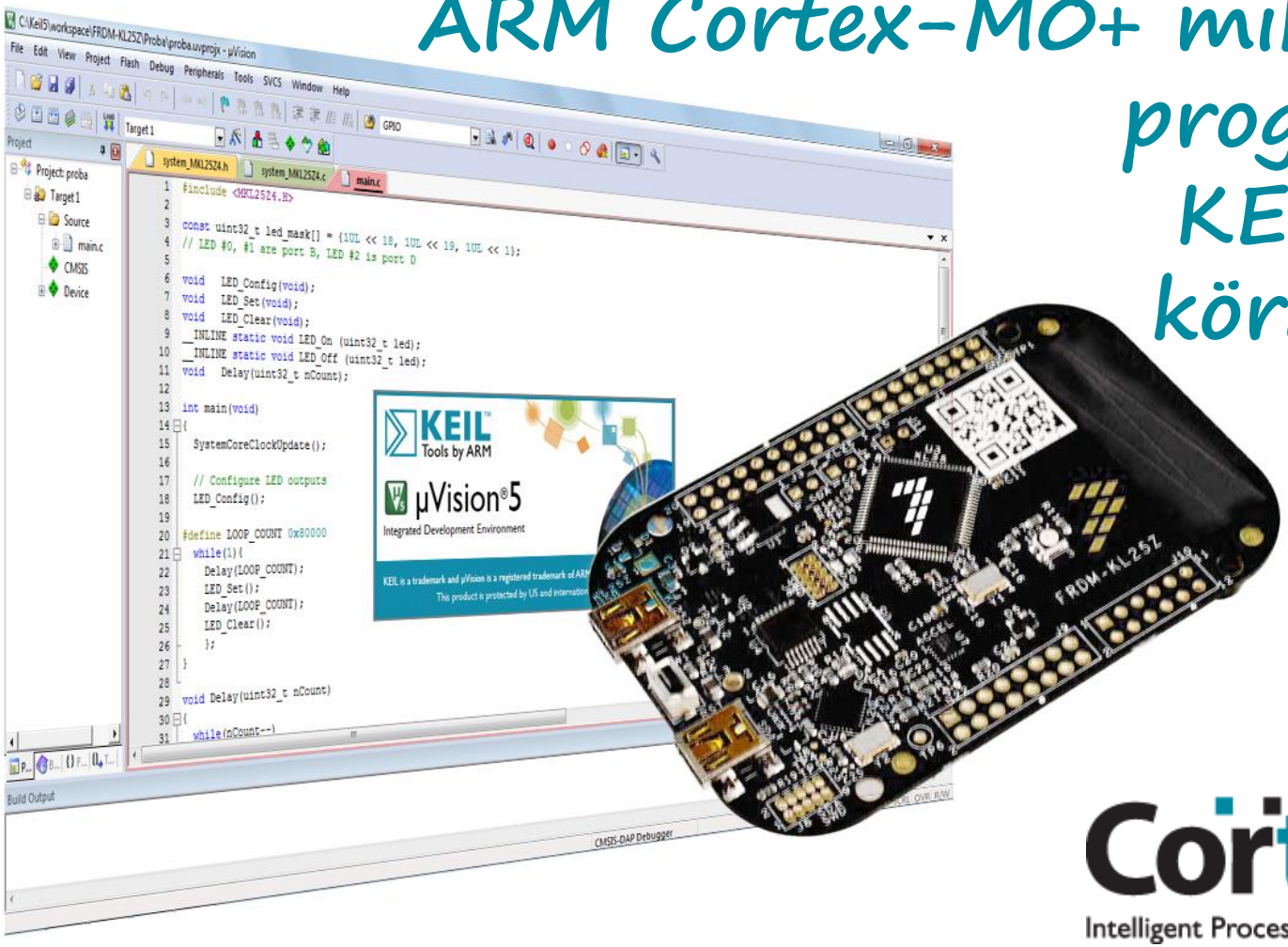
ARM Cortex-M0+ mikrovezérlő programozása KEIL MDK 5 környezetben



Cortex
Intelligent Processors by ARM®

13. DMA – közvetlen memória hozzáférés

Felhasznált anyagok, ajánlott irodalom

- ARM University Program: **Course/Lab Material for Teaching Embedded Systems/MCUs** (for the FRDM-KL25Z board)
- Freescale: **MKL25Z128VLK4 MCU datasheet**
- Freescale: **KL25 Sub-Family Reference Manual**
- Freescale: **FRDM-KL25Z User Manual**
- Freescale: **KLQRUG - Kinetis L Peripheral Module Quick Reference User's Guide**

Áttekintés

A DMA (közvetlen memória hozzáférés) lényege az, hogy memória területek, vagy memóriacímre leképezett perifériák között a CPU erőforrásainak igénybevétele nélkül végezhessünk adatátvitelt.

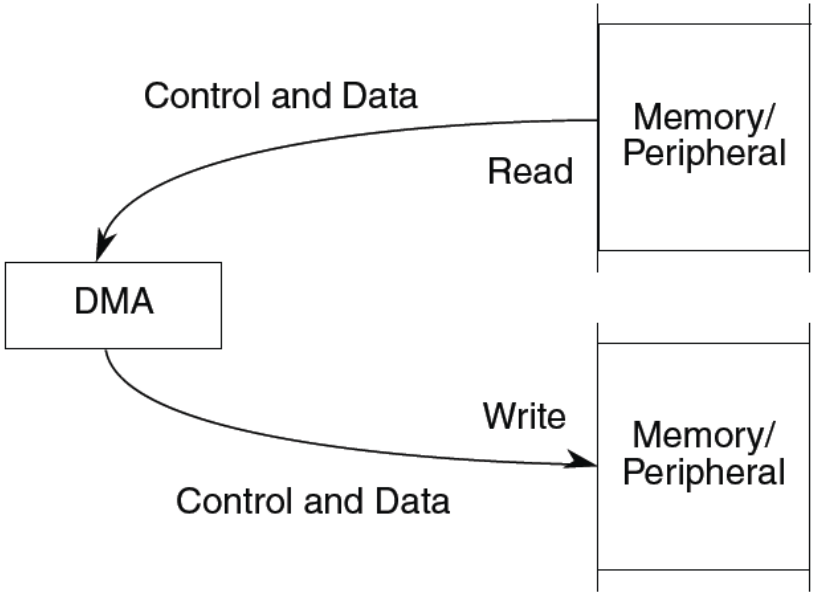
A cél a hatékonyság növelése, a CPU erőforrásainak felszabadítása más feladat szimultán végzésére.

Az érintett témakörök:

- **Működési elv**
- **DMA perifériák a Cortex-M0+ mikrovezérlőkben**
- **DMA alkalmazások**
  - Adatátvitel
  - Megszakítások kiváltása

Alapkoncepció

- **A DMA vezérlő egy speciális hardver periféria,** amely egy forráshelyről adatokat olvas és azokat egy célhelyre átmásolja

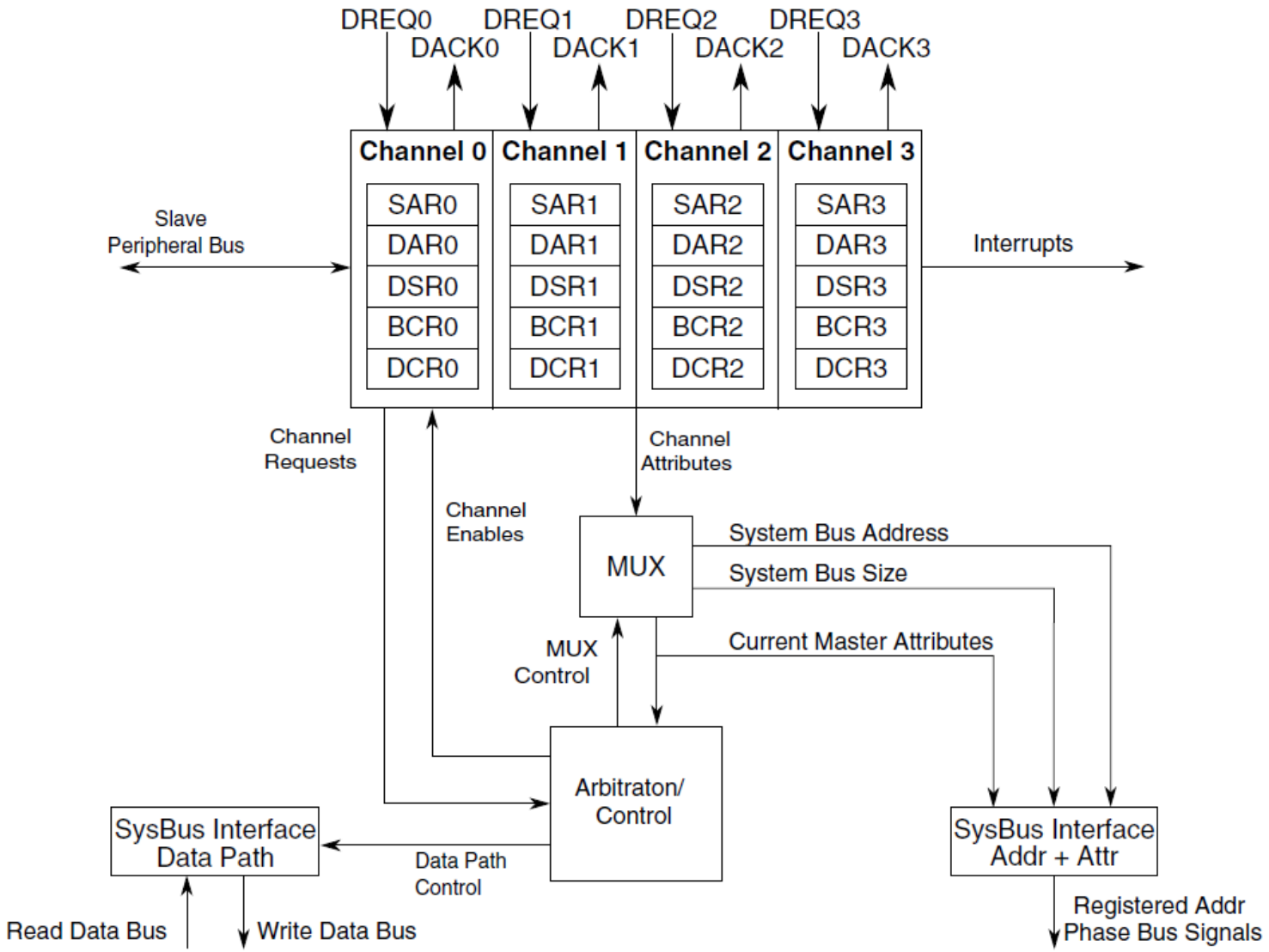


- **Különféle beállítási opciók**
  - Az átmásolandó adatok száma
  - A forrás és a cél címe lehet fix, vagy változó (azaz inkrementálódhat vagy dekrementálódhat minden átvitelnél)
  - Az adatelemek mérete
  - Mikor kezdődjön az átvitel
- **Műveletek**
  - **Inicializálás:** A **DMA** vezérlő konfigurálása
  - **Átvitel:** Az adat átmásolása
  - **Befejezés:** Az átviteli csatorna jelzi, hogy a tranzakció befejeződött

MKL25Z128VLK4 DMA jellemzők

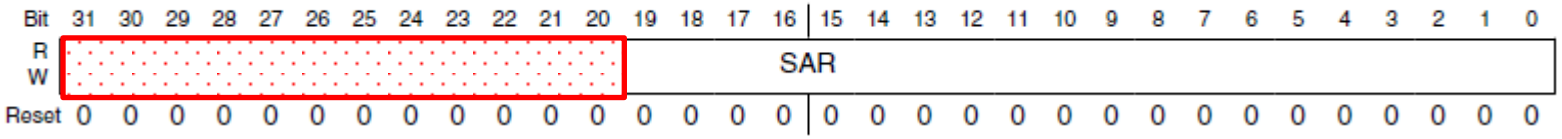
- 4 független csatorna
  - A 0. csatorna a legmagasabb prioritású
- 8-, 16- vagy 32-bites átvitel, a forrás és a cél adatszélessége különböző lehet
- Modulo címzés (gyűrűs tár)
- Hardveres vagy szoftveres indítás
- Az átvitel lehet folyamatos vagy periodikus (“ciklus-lopás”)
- Hardveres nyugtázó/készenlét jel
- DMA csatornák összefűzése

DMA vezérlő


DREQ0
DREQ1
DREQ2
DREQ3
DACK0
DACK1
DACK2
DACK3
Channel 0
Channel 1
Channel 2
Channel 3
SAR0
DAR0
DSR0
BCR0
DCR0
SAR1
DAR1
DSR1
BCR1
DCR1
SAR2
DAR2
DSR2
BCR2
DCR2
SAR3
DAR3
DSR3
BCR3
DCR3
Slave
Peripheral Bus
Interrupts
Channel
Requests
Channel
Attributes
Channel
Enables
MUX
System Bus Address
System Bus Size
Current Master Attributes
MUX
Control
Arbitraton/
Control
SysBus Interface
Data Path
Data Path
Control
SysBus Interface
Addr + Attr
Read Data Bus
Write Data Bus
Registered Addr
Phase Bus Signals

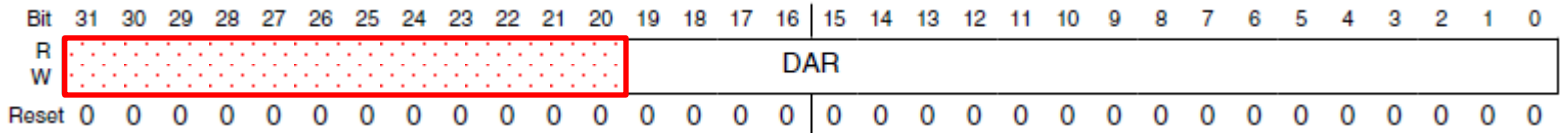
DMA_SARn forráscím regiszter

Address: 4000_8000h base + 100h offset + (16d × i), where i=0d to 3d



A 31 – 20. bitek tartalma csak 0x000x_xxxx, 0x1FFx_xxxx, 0x200x_xxxx, vagy 0x400x_xxxx lehet!

DMA_DARn cél címe regiszter



A 31 – 20. bitek tartalma csak 0x000x_xxxx, 0x1FFx_xxxx, 0x200x_xxxx, vagy 0x400x_xxxx lehet!

DMA_DSR_BCRn regiszter

Address: 4000_8000h base + 108h offset + (16d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CE	BES	BED	0	REQ	BSY	DONE	BCR							
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BCR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

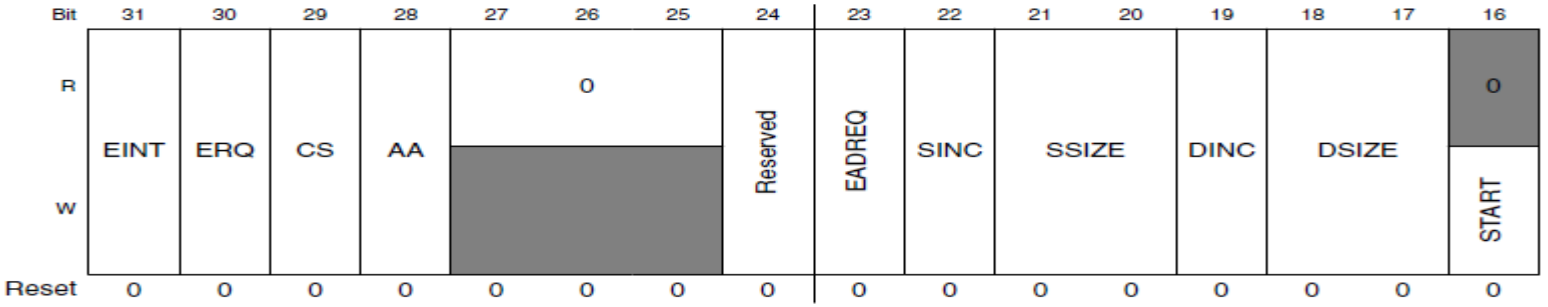
Állapotjelző bitek (a hibát vagy eseményt ’1’ jelzi)

- **CE:** Konfigurációs hiba
- **BES:** BUSZ hiba a forrásnál
- **BED:** BUSZ hiba a célnál
- **REQ:** Átvitel kérelem függőben (több végrehajtandó átvitel)
- **BSY:** DMA csatorna foglalt
- **DONE:** Csatorna átvitel befejeződött vagy hibára futott. Megszakításban törölni kell ezt a bitet!

Bájt számláló

- **BCR:** A hátralevő átviendő bájtok száma (1, 2, vagy 4-gyel csökken, átviteltől függően)

DMA vezérlő regiszter (DMA_DCRn)



16 – 31. bitek

EINT: Megszakítás átvitel végén engedélyezés

ERQ: Periféria által kezdeményezett átviteli kérelem engedélyezése

CS: Ciklusloás

0: Kapzsi mód - folyamatos DMA átvitel, amíg **BCR == 0** lesz

1: A DMA megosztja a buszt, kérelmenként csak egy átvitelt hajt végre

AA: Automatikus illesztés (ha a forrás és cél eltérő szélességű)

EADRQ: Aszinkron DMA kérelem (amikor a CPU alvás módban van) engedélyezése

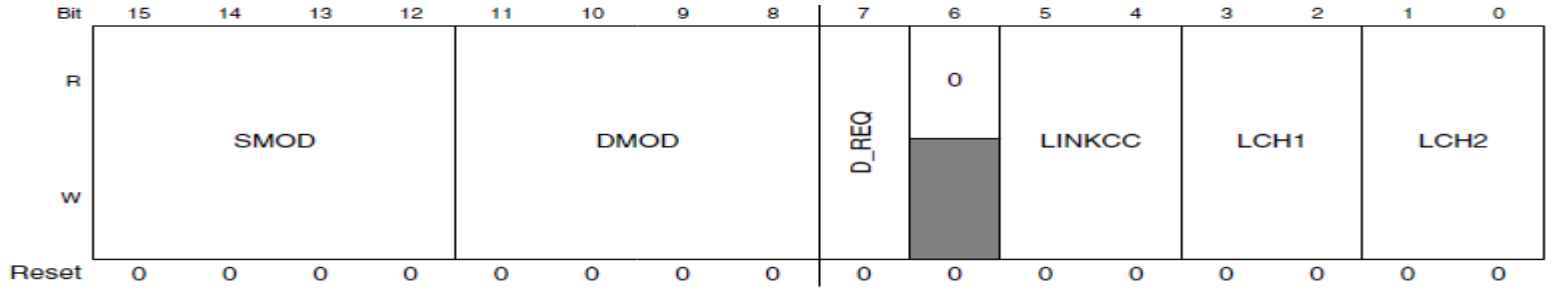
SINC/DINC: SAR/DAR növelése átvitelkor (1, 2 vagy 4 értékkel, mérettől függően)

SSIZE/DSIZE: Forrás/cél adatszélesség. Nem kell, hogy megegyezzenek.

00: long word (32 bit), **01:** byte (8 bit), **10:** word (16 bit)

START: Átvitel indítása **1** beírásakor (szoftveres indítás)

DMA vezérlő regiszter (DMA_DCRn)



0 – 15. bitek

SMOD, DMOD: Forrás/Cél címzés modulus – ha nem nulla, akkor gyűrűs adatbuffer kezelést támogat: körbefordulás 2^{n+3} bájtnál (bufferméret: 16 bájttól 64 kilobájtig). Ha nulla, akkor nincs gyűrűs adatbuffer támogatás

D_REQ: Ha 1, akkor **ERQ** automatikusan törlődik, amikor **BCR** nullára fut.

LINKCC: Engedélyezi, hogy ez a csatorna másik csatorna átvitelét indítsa

- 00: Letiltva
- 01: Kétféle esemény:
  - Az LCH1 indítása minden átvitelkor (kapzsi módban nem érvényesül!)
  - Az LCH2 indítása, amikor **BCR** nullára futott
- 10: Az LCH1 indítása minden átvitelkor (kapzsi módban nem érvényesül!)
- 11: Az LCH1 indítása, amikor **BCR** nullára futott

LCH1, LCH2: DMA csatorna sorszáma (0 to 3)

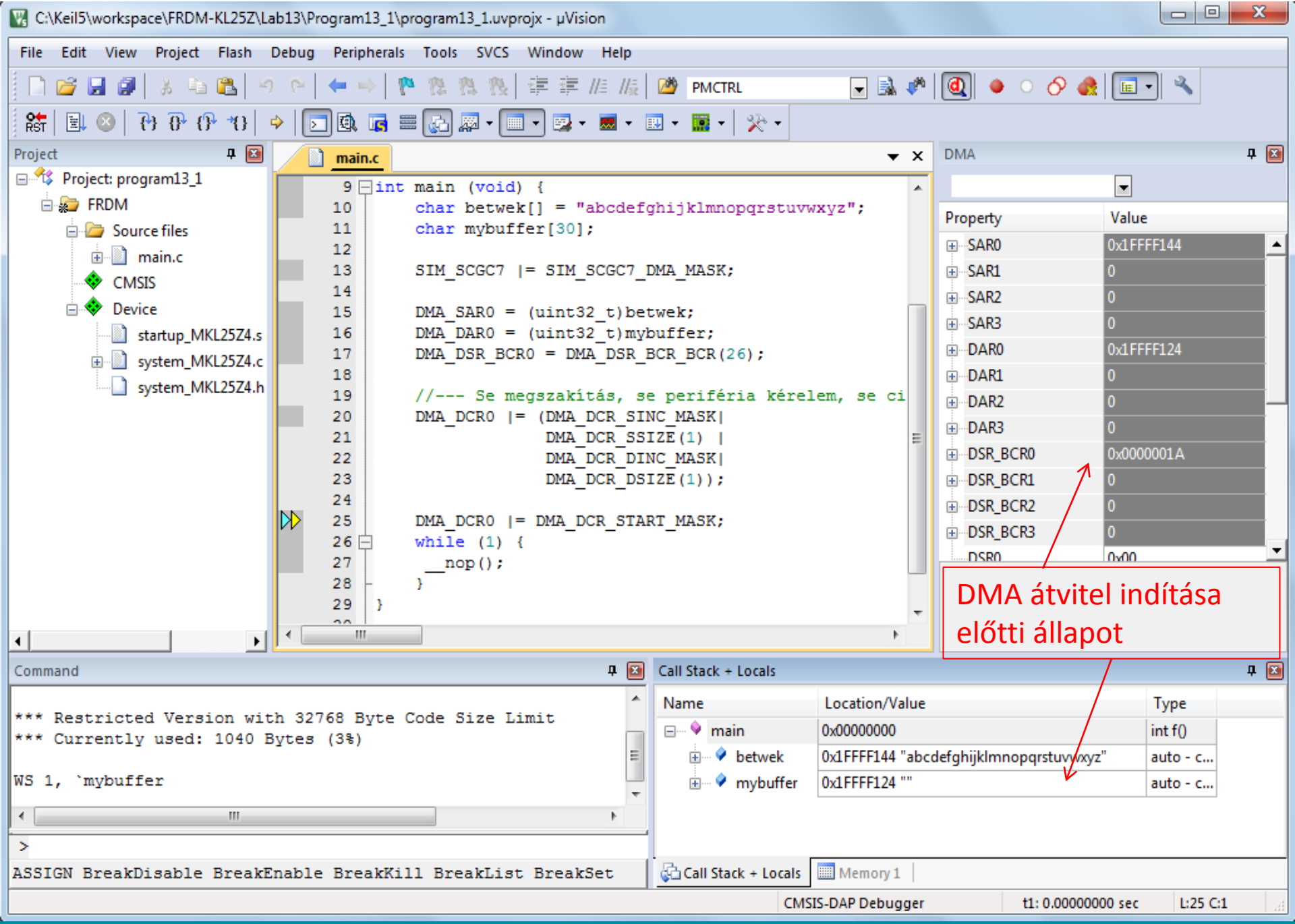
A DMA egyszerű használata

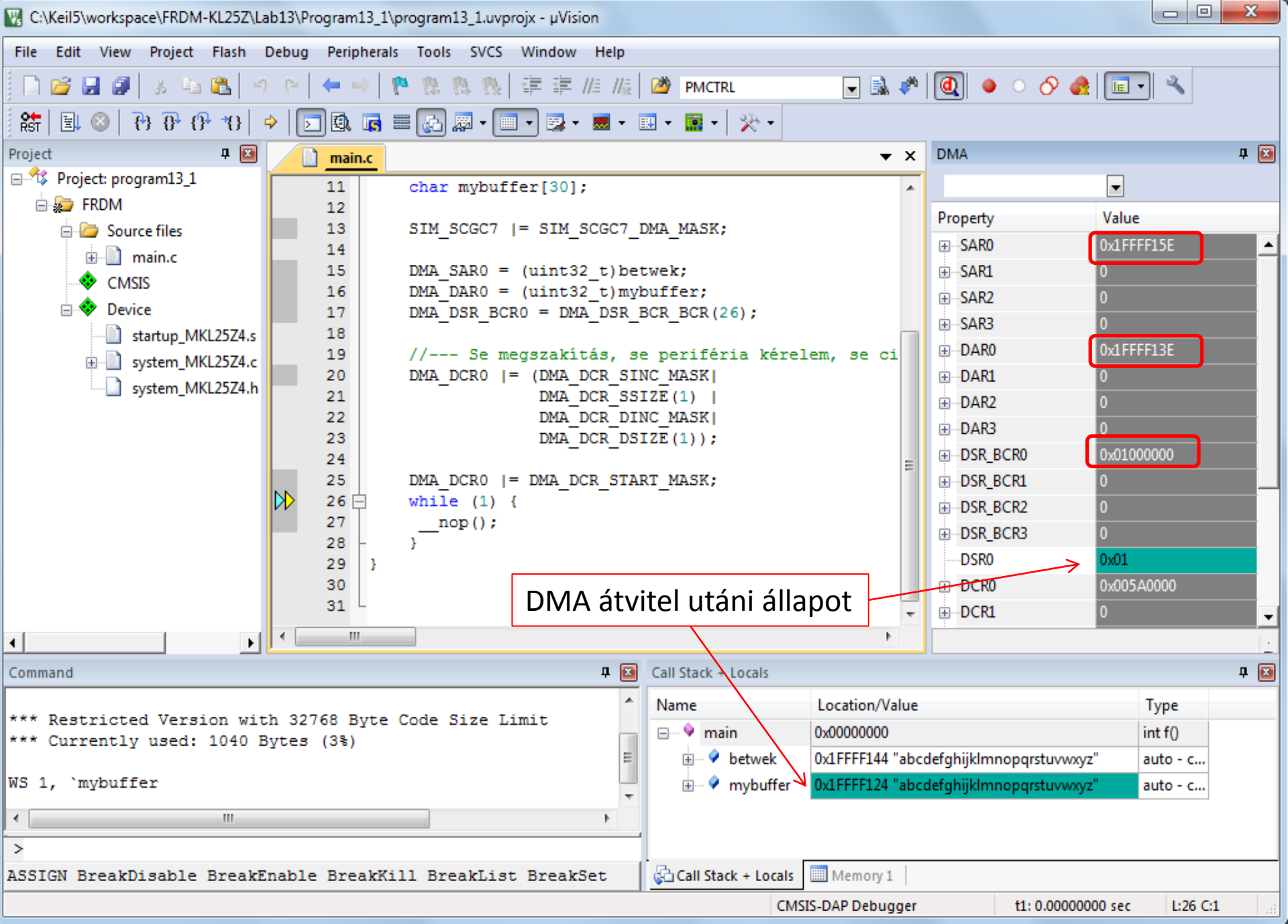
- Engedélyezzük a DMA modult (**SIM_SCGC7** regiszterben)
- A vezérlő regiszter konfigurálása
- **DMA_SARn** feltöltése a forrás címével
- **DMA_DARn** feltöltése a cél címével
- A **BCRn** bájtszámláló feltöltése az átvinni kívánt bájtok számával
- Töröljük a **DSRn[DONE]** bitet
- Elindítjuk az átvitelt a **DCRn[START]** bit 1-be állításával
- Az átvitel végére várunk:
  - **DMAn_IRQ** megszakítás keletkezik, ha **DCRn[EINT] = 1**
  - Lekérdezzük a **DSRn[DONE]** bitet

Program13_1: egyszerű adatmozgatás

Egyszerű adatmozgatás a memóriában DMA felhasználásával.
A DMA átvitelt szoftveresen indítjuk.
A DMA átvitel végét nem figyeljük a programban.
A program vizsgálatához a Keil MDK5 IDE hardveres nyomkövetés üzemmódját használjuk

```
#include <MKL25Z4.H>

int main (void) {
    char betwek[] = "abcdefghijklmnopqrstuvwxyz";     // latin ABC
    char mybuffer[30];                                // tárterület az átvitelhez

    SIM_SCGC7 |= SIM_SCGC7_DMA_MASK;                  // DMA engedélyezése

    DMA_SAR0 = (uint32_t)betwek;                      // Forráscím megadása
    DMA_DAR0 = (uint32_t)mybuffer;                    // Célterület kezdőcíme
    DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(26);               // 26 bájtos az átvitel

    //--- Se megszakítás, se periféria kérelem, se ciklusloPás...
    DMA_DCR0 |= (DMA_DCR_SINC_MASK|                   // A forráscímet inkrementálni kell
                 DMA_DCR_SSIZE(1) |                   // Forrásméret: 8 bites adatok
                 DMA_DCR_DINC_MASK|                   // A cél címét inkrementálni kell
                 DMA_DCR_DSIZE(1));                   // A célterület mérete: 8 bites adatok

    DMA_DCR0 |= DMA_DCR_START_MASK;                   // DMA átvitel indítása
    while (1) {
     __nop();
    }
}
```


C:\Keil5\workspace\FRDM-KL25Z\Lab13\Program13_1\program13_1.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PMCTRL
Project
Project: program13_1
FRDM
Source files
main.c
CMSIS
Device
startup_MKL25Z4.s
system_MKL25Z4.c
system_MKL25Z4.h
main.c
9 int main (void) {
10 char betwek[] = "abcdefghijklmnopqrstuvwxyz";
11 char mybuffer[30];
12
13 SIM_SCGC7 |= SIM_SCGC7_DMA_MASK;
14
15 DMA_SAR0 = (uint32_t)betwek;
16 DMA_DAR0 = (uint32_t)mybuffer;
17 DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(26);
18
19 //--- Se megszakítás, se periféria kérelem, se ci
20 DMA_DCR0 |= (DMA_DCR_SINC_MASK|
21 DMA_DCR_SSIZE(1) |
22 DMA_DCR_DINC_MASK|
23 DMA_DCR_DSIZE(1));
24
25 DMA_DCR0 |= DMA_DCR_START_MASK;
26 while (1) {
27 __nop();
28 }
29 }
DMA
Property
Value
SAR0 0x1FFFF144
SAR1 0
SAR2 0
SAR3 0
DAR0 0x1FFFF124
DAR1 0
DAR2 0
DAR3 0
DSR_BCR0 0x0000001A
DSR_BCR1 0
DSR_BCR2 0
DSR_BCR3 0
DMA átvitel indítása előtti állapot
Command
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 1040 Bytes (3%)
WS 1, `mybuffer
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Name
Location/Value
Type
main 0x00000000 int f()
betwek 0x1FFFF144 "abcdefghijklmnopqrstuvwxyz" auto - c...
mybuffer 0x1FFFF124 "" auto - c...
Call Stack + Locals
Memory 1
CMSIS-DAP Debugger
t1: 0.00000000 sec
L:25 C:1

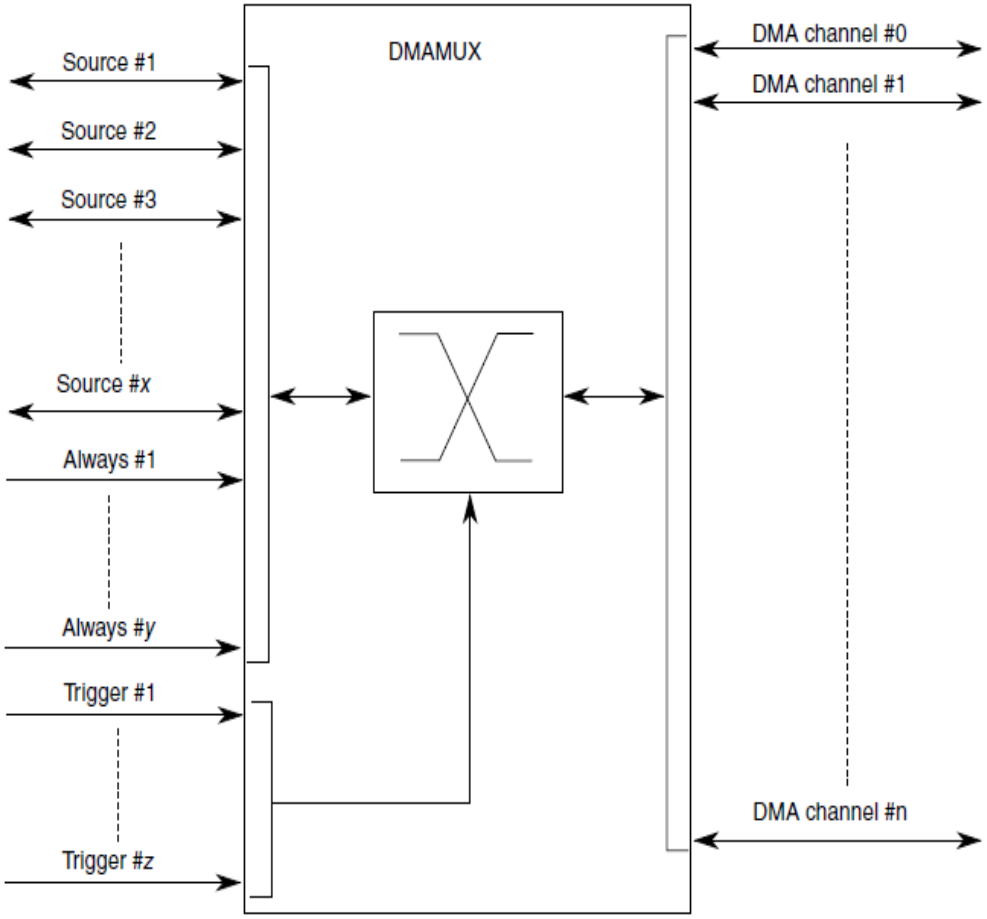

C:\Keil5\workspace\FRDM-KL25Z\Lab13\Program13_1\program13_1.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PMCTRL
Project
Project: program13_1
FRDM
Source files
main.c
CMSIS
Device
startup_MKL25Z4.s
system_MKL25Z4.c
system_MKL25Z4.h
main.c
11 char mybuffer[30];
12
13 SIM_SCGC7 |= SIM_SCGC7_DMA_MASK;
14
15 DMA_SAR0 = (uint32_t)betwek;
16 DMA_DAR0 = (uint32_t)mybuffer;
17 DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(26);
18
19 //--- Se megszakítás, se periféria kérelem, se ci
20 DMA_DCR0 |= (DMA_DCR_SINC_MASK|
21 DMA_DCR_SSIZE(1) |
22 DMA_DCR_DINC_MASK|
23 DMA_DCR_DSIZE(1));
24
25 DMA_DCR0 |= DMA_DCR_START_MASK;
26 while (1) {
27 __nop();
28 }
29 }
30
31
DMA
Property
Value
SAR0 0x1FFFF15E
SAR1 0
SAR2 0
SAR3 0
DAR0 0x1FFFF13E
DAR1 0
DAR2 0
DAR3 0
DSR_BCR0 0x01000000
DSR_BCR1 0
DSR_BCR2 0
DSR_BCR3 0
DSR0 0x01
DCR0 0x005A0000
DCR1 0
DMA átvitel utáni állapot
Command
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 1040 Bytes (3%)
WS 1, `mybuffer
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Call Stack + Locals
Name
Location/Value
Type
main 0x00000000 int f()
betwek 0x1FFFF144 "abcdefghijklmnopqrstuvwxyz" auto - c...
mybuffer 0x1FFFF124 "abcdefghijklmnopqrstuvwxyz" auto - c...
Call Stack + Locals
Memory 1
CMSIS-DAP Debugger
t1: 0.00000000 sec
L:26 C:1

DMA átvitel indítása perifériákkal

Bit	7	6	5	4	3	2	1	0
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFGn bitkiosztása

- A DMA átvitelt periféria esemény is indíthatja
- Triggereléskor a DMA végezhet:
  - Egy átvitelt (ciklusolás mód)
  - Teljes átvitel, amíg BCR == 0 lesz (folyamatos mód)
- DMA Multiplexer (DMAMUX)
  - Kiválasztja, hogy mi legyen a DMA csatorna indítási eseménye
- Minden DMA csatornának van egy **DMAMUX_CHCFGn** konfigurációs regisztere
  - **ENBL:** Engedélyezi a csatornát
  - **TRIG:** Engedélyezi a DMA csatorna periodikus triggerelését (CH0 és CH1)
  - **SOURCE:** triggerforrás kijelöllése



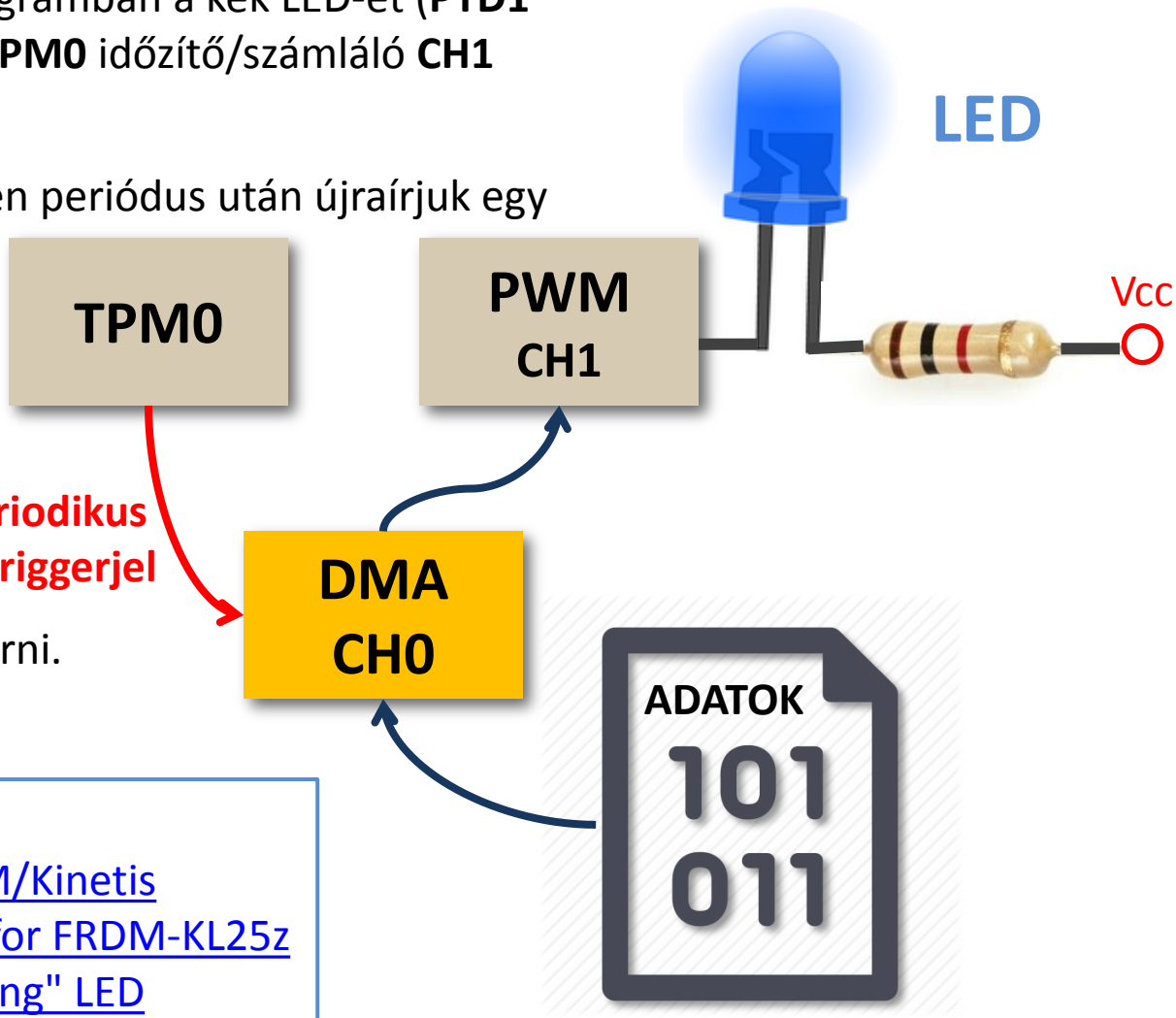
Trigger jelforrások

A DMA átvitelt indító hardveres jelforrások mikrovezérlő specifikusak.
A **KL25 Sub-Family Reference Manual** 3. fejezetének *Chip Configuration, Section 3.4.8.1 DMA MUX Request Sources* című szekciójában sorolja fel a DMA csatornákhoz rendelhető jelforrásokat.

Sorszám	Modul	Leírás
0	-	Letiltva
2-7	UART0,1, 2	Vétel (Rx) és Adás (Tx)
16-19	SPI0, 1	Vétel (Rx) és Adás (Tx)
22-23	I^2C0, 1	
24-29	TPM0	0-5. csatornák
32-35	TPM1-2	0-1. csatornák
40	ADC0	
42	CMP0	
45	DAC0	
49-53	Portvezérlő Modulok	Port A-E
54-56	TPM0-2	Túlcsordulás
57	TSI	
60-63	DMAMUX	Mindig engedélyezve

PWM kitöltés vezérlése DMA-val

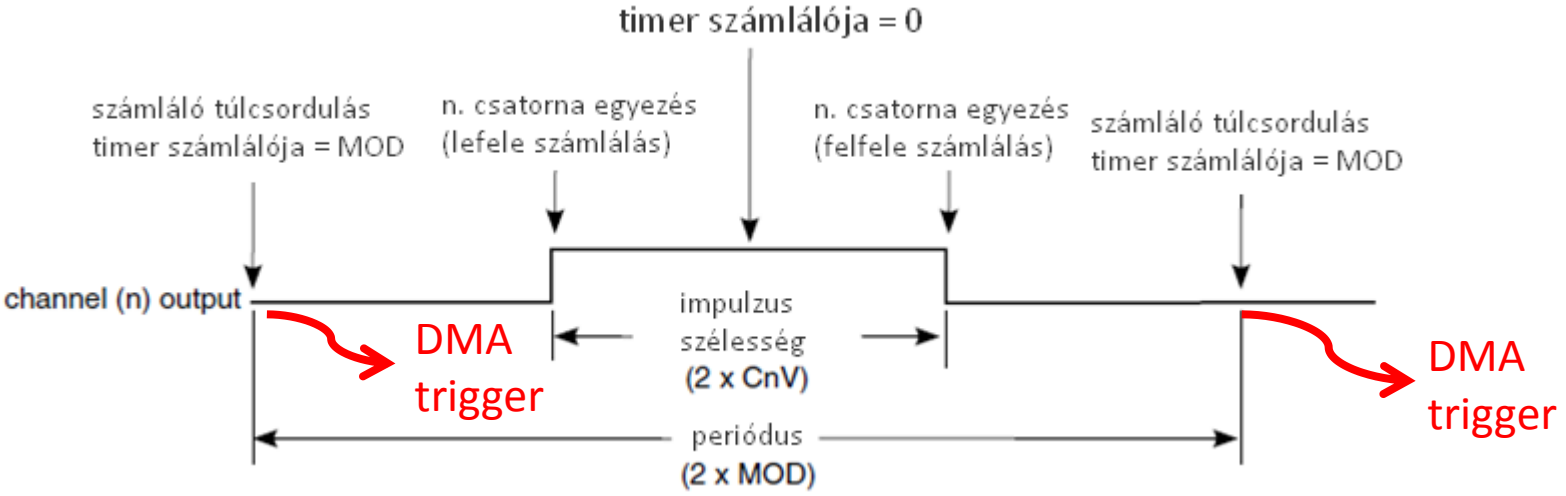
- A **Program13_2** mintaprogramban a kék LED-et (**PTD1** kivezetésre van kötve) a **TPM0** időzítő/számláló **CH1** PWM csatornája vezérli.
- A kitöltési tényezőt minden periódus után újraírjuk egy 256 elemű adattömbből.
- A DMA aktiválásához **TPM0** túlcsordulási eseménye ad triggerjelet.
- A bájtszámláló nullára futásakor elegendő a DMA bájtszámlálóját újraírni.



Felhasznált források:

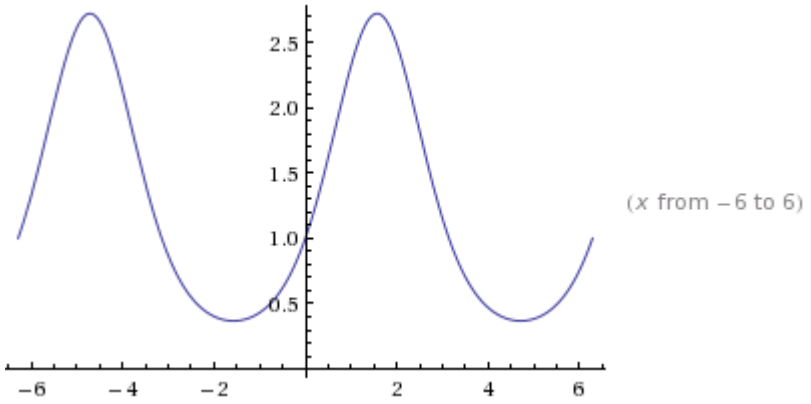
- PWM with DMA on ARM/Kinetis
- PIT-ADC-DMA example for FRDM-KL25z
- Re-creating the "breathing" LED

Program13_2



- Mivel „középre igazított” PWM jelet állítunk elő, a DMA triggerelését nem a csatorna egyezésnél, hanem a számláló túlcsordulásakor keltkező eseményre bízzuk.
- A táblázatban tárolt adatok a „lélegző LED”-et megközelítő függvény előre kiszámolt adatai egy periódusra. A függvényt úgy transzformáltuk, hogy a kitöltést beállító adatok értéke 0 és kb. 44 000 közötti szám legyen.

$$f(x) = e^{\sin(x)}$$



Program13_2

A programban a CPU frekvencia 48 MHz, a TPM0 timer előosztója ezt 6 MHz-re osztja le. A CH1 PWM csatorna modulo számlálója kétszer 50 000-et számlál, a PWM frekvencia így 60 Hz. A táblázatbeli adatok periódusa 256, a LED tehát kb. 4.3 s periódusidővel „lélegzik”.

```
#include <MKL25Z4.h>

uint16_t dmaSrc[] __attribute__((at(0x20001000))) = {
  11620, 12077, 12544, 13023, 13513, 14014, 14525, 15047, 15580, 16123, 16676, 17239, 17811, 18393, 18984, 19583,
  20190, 20805, 21427, 22056, 22690, 23330, 23975, 24624, 25277, 25932, 26589, 27246, 27904, 28562, 29217, 29870,
  30519, 31164, 31802, 32435, 33059, 33674, 34280, 34874, 35456, 36025, 36579, 37118, 37641, 38145, 38632, 39098,
  39543, 39967, 40369, 40747, 41100, 41428, 41731, 42006, 42255, 42476, 42668, 42832, 42966, 43071, 43146, 43191,
  43206, 43191, 43146, 43071, 42966, 42832, 42668, 42476, 42255, 42006, 41731, 41428, 41100, 40747, 40369, 39967,
  39543, 39098, 38632, 38145, 37641, 37118, 36579, 36025, 35456, 34874, 34280, 33674, 33059, 32435, 31802, 31164,
  30519, 29870, 29217, 28562, 27904, 27246, 26589, 25932, 25277, 24624, 23975, 23330, 22690, 22056, 21427, 20805,
  20190, 19583, 18984, 18393, 17811, 17239, 16676, 16123, 15580, 15047, 14525, 14014, 13513, 13023, 12544, 12077,
  11620, 11174, 10740, 10316,  9904,  9502,  9111,  8731,  8362,  8003,  7655,  7316,  6988,  6670,  6362,  6064,
   5775,  5495,  5225,  4963,  4710,  4466,  4231,  4004,  3784,  3573,  3369,  3173,  2985,  2804,  2629,  2462,
   2301,  2147,  2000,  1858,  1723,  1594,  1471,  1353,  1241,  1135,  1034,   938,   848,   762,   681,   606,
    535,   469,   407,   350,   298,   249,   206,   166,   131,   100,    74,    51,    33,    18,     8,     2,
      0,     2,     8,    18,    33,    51,    74,   100,   131,   166,   206,   249,   298,   350,   407,   469,
    535,   606,   681,   762,   848,   938,  1034,  1135,  1241,  1353,  1471,  1594,  1723,  1858,  2000,  2147,
   2301,  2462,  2629,  2804,  2985,  3173,  3369,  3573,  3784,  4004,  4231,  4466,  4710,  4963,  5225,  5495,
   5775,  6064,  6362,  6670,  6988,  7316,  7655,  8003,  8362,  8731,  9111,  9502,  9904, 10316, 10740, 11174
};

//--- DMA megszakítások kiszolgálása ----------
void DMA0_IRQHandler(void) {
  DMA_DSR_BCR0 |= DMA_DSR_BCR_DONE_MASK;        // DMA Done jelző törlése
  DMA_DSR_BCR0 |= DMA_DSR_BCR_BCR(512);         // Új átvitel méretének megadása
}
```

```
int main (void) {
    SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK;          // PORTD engedélyezése
    PORTD->PCR[1] = PORT_PCR_MUX(4);             // PTD1 legyen TPM0 kimenet
    SIM->SCGC6 |= SIM_SCGC6_TPM0_MASK;           // A TPM0 modul engedélyezése
    SIM->SOPT2 = SIM_SOPT2_TPMSRC(1) |
                 SIM_SOPT2_PLLFLLSEL(1);         // MCGPLLCLK/2 legyen TPM0 órajele
    TPM0->SC = 0;                                // TPM0 letiltása a konfiguráláshoz
    TPM0->CONTROLS[1].CnSC = 0x20 | 0x04;        // Középre-igazított, alacsony kimenet, DMA eng.
    TPM0->MOD = 50000;                           // A periódusidő 60 Hz (1:8 előosztással)
    TPM0->CONTROLS[1].CnV = 25000;               // CH1-et 50 %-os kitöltésre állítjuk
    TPM0->SC = TPM_SC_DMA(1)    |                // DMA kérelem kiadása túlcsorduláskor
               TPM_SC_CPWMS(1)  |                // Középre igazított PWM jel
               TPM_SC_CMOD(1)   |                // Számlálás belső órajelre
               TPM_SC_PS(3);                     // PS = 011, 1:8 előosztás
    SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;          // DMAMUX engedélyezése
    DMAMUX0_CHCFG0 = 0x00;                       // DMAMUX 0. csatorna leállítás konfiguráláshoz
    SIM->SCGC7 |= SIM_SCGC7_DMA_MASK;            // DMA engedélyezése
    DMA_SAR0 = (uint32_t)dmaSrc;                 // Forráscím megadása
    DMA_DAR0 = (uint32_t)&TPM0_C1V;              // Célterület kezdőcíme
    DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(512);         // 512 bájtos az átvitel
    DMA_DCR0 |= (DMA_DCR_EINT_MASK|              // Megszakítás engedélyezése
                 DMA_DCR_ERQ_MASK |              // Periféria DMA kérelem engedélyezése
                 DMA_DCR_CS(1)    |              // Ciklusloás mód
                 DMA_DCR_SMOD(6)  |              // Forráscím 512 bájt után körbefordul
                 DMA_DCR_SINC(1)  |              // A forráscímet inkrementálni kell
                 DMA_DCR_SSIZE(2) |              // Forrásméret: 16 bites adatok
                 DMA_DCR_DINC(0)  |              // A cél címét nem inkrementáljuk
                 DMA_DCR_DSIZE(2));              // A célterület mérete: 16 bites adatok
    DMAMUX0_CHCFG0 |= DMAMUX_CHCFG_ENBL_MASK |   // Normál aktiválás (nincs PIT triggerelés)
                      DMAMUX_CHCFG_SOURCE(54);   // TPM0 túlcsordulása indítja a DMA átvitelt
    NVIC_EnableIRQ(DMA0_IRQn);                   // DMA megszakítások engedélyezése
    __enable_irq();                              // Megszakítások globális engedélyezése
    while(1) {
      __nop();
    }
}
```

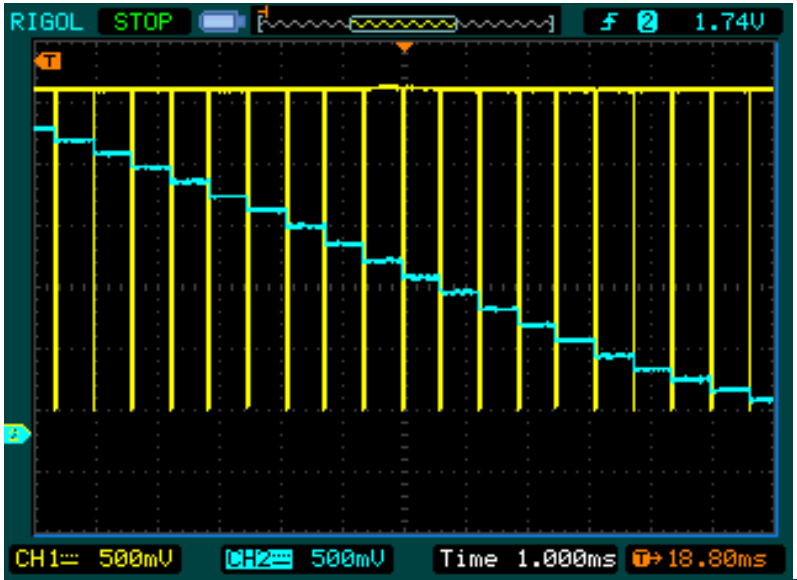
Megszakítás helyett DMA: DAC Playback

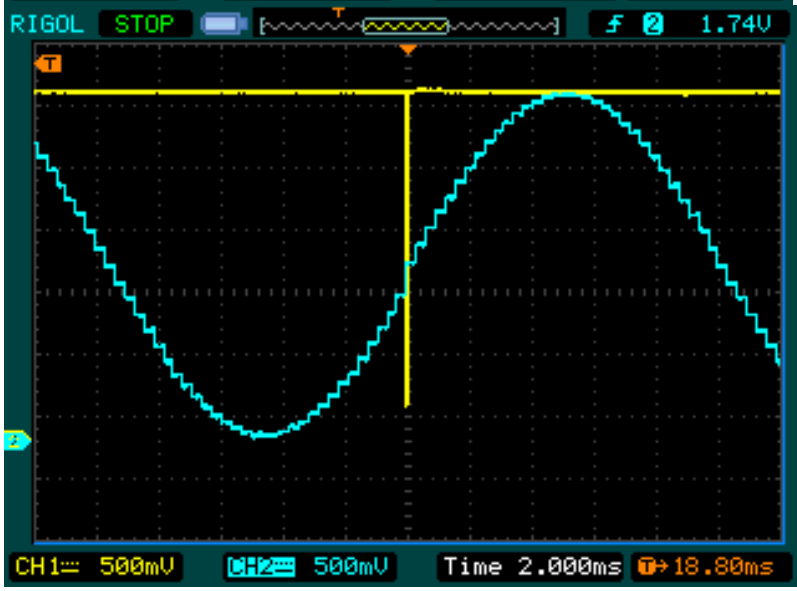
- TPM0 túlcsordul, amikor a DAC adatregiszterét frissíteni kell
- Amikor TPM0 DMA átvitelt kezdeményez:
  - Egy 10 bites adatelem kiküldésre kerül a hullámtáblából (forrás) a DAC adatregiszterébe
  - A forráscím inkrementálódik, a bájtszámláló pedig dekrementálódik
- Amikor a DMA az utolsó átvitelt befejezi
  - Megszakítást generál
  - A megszakításban töröljük a jelzőbitet és új adatcsomag küldést írunk elő
- DMA konfigurálás
  - Forrás és cél adatméret = 2 bájt: SSIZE = DSIZE = 2
  - Engedélyezzük a perifériás átviteli kérelmeket: ERQ = 1
  - Hardveres triggerelést állítunk be
  - Egy átvitel periféria kérelmenként, tehát a „cikluslopás” módot állítjuk be: CS = 1
  - Két bájnyi adat átvitelenként: BCR = 2*n bájt (ahol n az adattáblában levő adatok száma)
  - Átvitel után léptetni kell a forrás címét: SINC = 1
  - Nem léptetjük a cél címét: DINC = 0
- TPM0 Configuration
  - Nem engedélyezzük az időzítő megszakításait
  - TPM0 így is generál átviteli kérelmeket minden túlcsorduláskor

Mi a programnak egy módosított változatát fogjuk elkészíteni: TPM0 helyett PIT triggerel.

Teljesítmények összehasonlítása

- Jelek
  - Sárga: Megszakítás fut, míg a jel alacsony
  - Kék: DAC kimenet (PTE30 kivezetés)
- DMA nélkül: Megszakítás mintánként
  - 4.7 µs 620 mikroszekundumonként
  - 0.758%-át veszi igénybe a CPU időnek
- DMA-val: Megszakítás jelciklusonként
  - 5.0 µs 20 milliszekundumonként
  - 0.025%-át veszi igénybe a CPU időnek
- Mennyiben hasznos ez?
  - CPU időt takarít meg
  - Csökkenti az időzítési pontatlanságot (pl. megszakítás tiltása miatt)
  - A CPU tovább alhat és, ritkábban kell felébreszteni (20 milliszekundumonként a 620 mikroszekundum helyett)





Program13_3: PIT-DMA-DAC

```
/* Program13_4
 *
 * DAC hullámtábla lejátszása DMA felhasználásával.
 * A DMA átvitelt a PIT 0. csatornája indítja
 *
 */

#include <MKL25Z4.h>

uint16_t dmaSrc[] __attribute__((at(0x20001000))) =
{ 2047, 2247, 2446, 2641, 2830, 3011, 3184, 3345,
  3494, 3629, 3749, 3852, 3938, 4005, 4054, 4084,
  4094, 4084, 4054, 4005, 3938, 3852, 3749, 3629,
  3494, 3345, 3184, 3011, 2830, 2641, 2446, 2247,
  2047, 1846, 1647, 1452, 1263, 1082,  909,  748,
   599,  464,  344,  241,  155,   88,   39,    9,
     0,    9,   39,   88,  155,  241, 344,   464,
   599,  748,  909, 1082, 1263, 1452, 1647, 1846
};

uint32_t bus_frequency(void) {
    return (SystemCoreClock / (((SIM->CLKDIV1 & SIM_CLKDIV1_OUTDIV4_MASK)
    >> SIM_CLKDIV1_OUTDIV4_SHIFT) + 1));
}

//--- DMA megszakítások kiszolgálása --------------
void DMA0_IRQHandler(void) {
    DMA_DSR_BCR0 |= DMA_DSR_BCR_DONE_MASK;          // DMA Done jelző törlése
    DMA_DSR_BCR0 |= DMA_DSR_BCR_BCR(128);           // Új átvitel méretének megadása
}
```

Egy színuszhullám
64 db. 12 bites adattal

Program13_3: PIT-DMA-DAC

- A DAC kimenőjelének engedélyezéséhez engedélyeznünk és analóg módba kell állítanunk a **PTE30** kivezetést.
- A DAC modult engedélyezés után nem bufferelt, „azonnali” módba állítjuk.
- A PIT modul engedélyezése után törölnünk kell a modul letiltójelét is (PIT->MCR).
- Az újratöltési regiszterbe a minták lejátszása közt eltölteni kívánt buszciklusok számát kell megadni. `bus_frequency()/64/1000` megadása esetén 1 ms alatt küldjük ki a teljes színuszhullámot, azaz 1000 Hz-es színuszjel jelenik meg a kimeneten.
- A PIT számlálót elindítjuk, de nem engedélyezzük, hogy megszakítást kérjen. A DMA triggerelést így is elvégzi

```
int main (void) {
    //--- DAC konfigurálása ----------------------
    SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;            // PORTE engedélyezése
    PORTE->PCR[30] = 0;                            // PTE30 analóg módban legyen
    SIM->SCGC6 |= SIM_SCGC6_DAC0_MASK;             // DAC0 engedélyezése
    DAC0->C1 = 0;                                  // Bufferelt mód letiltása
    DAC0->C0 = DAC_C0_DACEN_MASK |                 // DAC0 bekapcsolása
               DAC_C0_DACRFS(1);                   // VREF = VDDA

    //--- PIT 0. csatorna konfigurálása -----------
    SIM->SCGC6 |= SIM_SCGC6_PIT_MASK;              // PIT modul engedélyezése
    PIT_MCR = 0;                                   // PIT letiltás feloldása
    PIT_LDVAL0 = bus_frequency()/64/1000;          // 64 000 minta/s
    PIT_TCTRL0 = PIT_TCTRL_TEN_MASK;               // Számláló indítás, nincs megszakítás
```

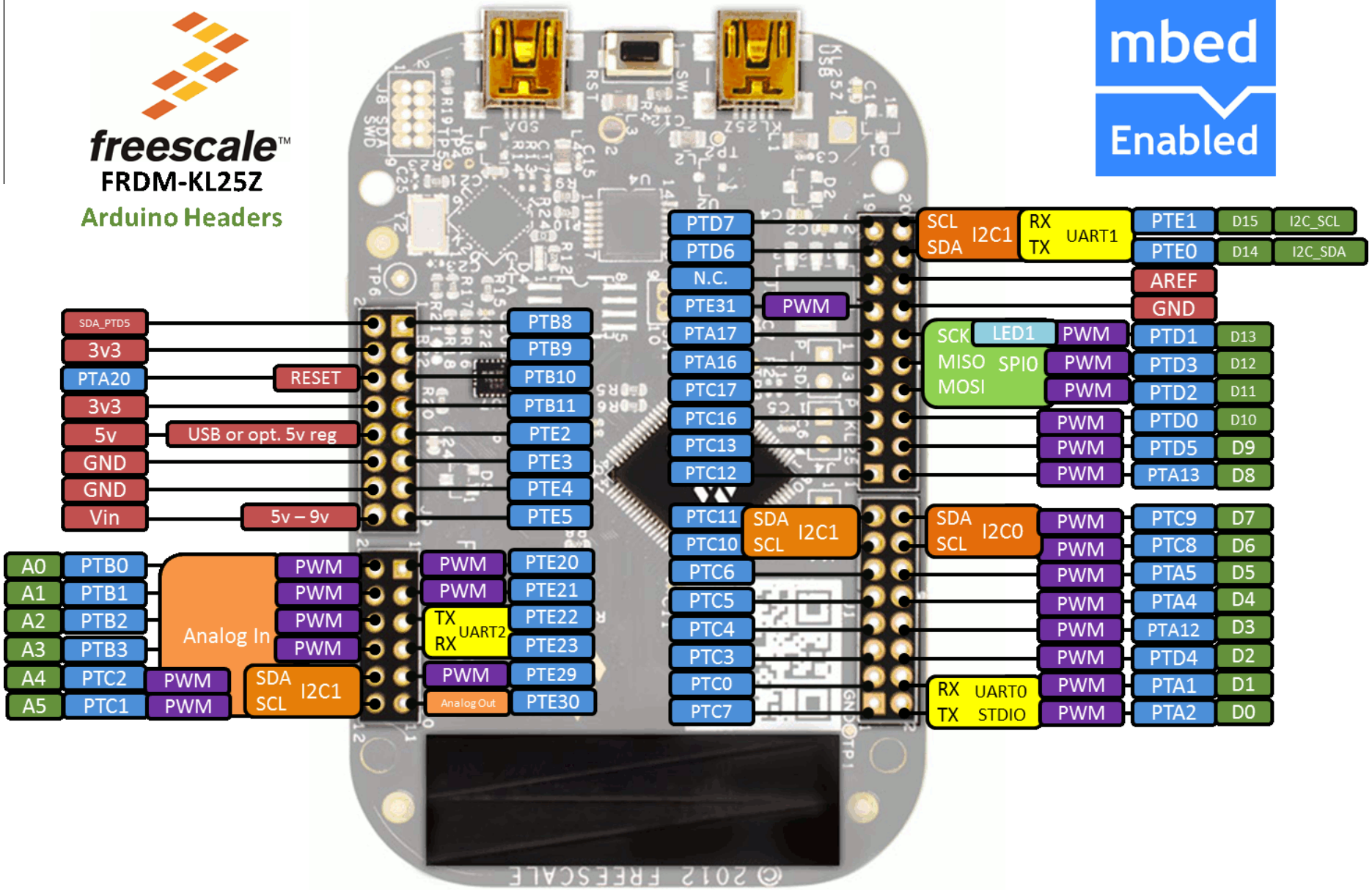
A 0. DMA csatornát most triggerelés módba állítjuk. A DMAMUX jelforrás most a 60 sorszámú Always0 (mindig engedélyezve) jel lesz, de 1-be állítjuk a Trigger bitet a DMAMUX_CHCFG0 regiszterben.

```
//--- DMAMUX és DMA CH0 konfigurálása ---------
    SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;          // DMAMUX engedélyezése
    DMAMUX0_CHCFG0 = 0x00;                       // DMAMUX 0. csatorna leállítás konfiguráláshoz

    SIM->SCGC7 |= SIM_SCGC7_DMA_MASK;            // DMA engedélyezése
    DMA_SAR0 = (uint32_t)dmaSrc;                 // Forráscím megadása
    DMA_DAR0 = (uint32_t)&DAC0_DAT0L;            // DAC0 adatregiszter a cél
    DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(128);         // 128 bájtos az átvitel

    DMA_DCR0 = (DMA_DCR_EINT_MASK|               // Megszakítás engedélyezése
                DMA_DCR_ERQ_MASK |               // Periféria DMA kérelem engedélyezése
                DMA_DCR_CS(1)    |               // Ciklusolpás mód
                DMA_DCR_SMOD(4)  |               // Forráscím 128 bájt után körbefordul
                DMA_DCR_SINC(1)  |               // A forráscímet inkrementálni kell
                DMA_DCR_SSIZE(2) |               // Forrásméret: 16 bites adatok
                DMA_DCR_DINC(0)  |               // A cél címét nem inkrementáljuk
                DMA_DCR_DSIZE(2));               // A célterület mérete: 16 bites adatok

    DMAMUX0_CHCFG0 = DMAMUX_CHCFG_ENBL_MASK |    // 0. csatorna engedélyezése
                     DMAMUX_CHCFG_TRIG_MASK |    // PIT triggerelés
                     DMAMUX_CHCFG_SOURCE(60);    // Mindig engedélyezve

    //--- Megszakítások engedélyezése -------------
    NVIC_EnableIRQ(DMA0_IRQn);                   // DMA megszakítások engedélyezése
    __enable_irq();                              // Megszakítások globális engedélyezése

    while(1) {
      __nop();
      __nop();
    }
}
```

A főprogramban nincs semmi dolgunk!

freescale™
FRDM-KL25Z
Arduino Headers

mbed Enabled



freescale™

FRDM-KL25Z

Additional Peripherals