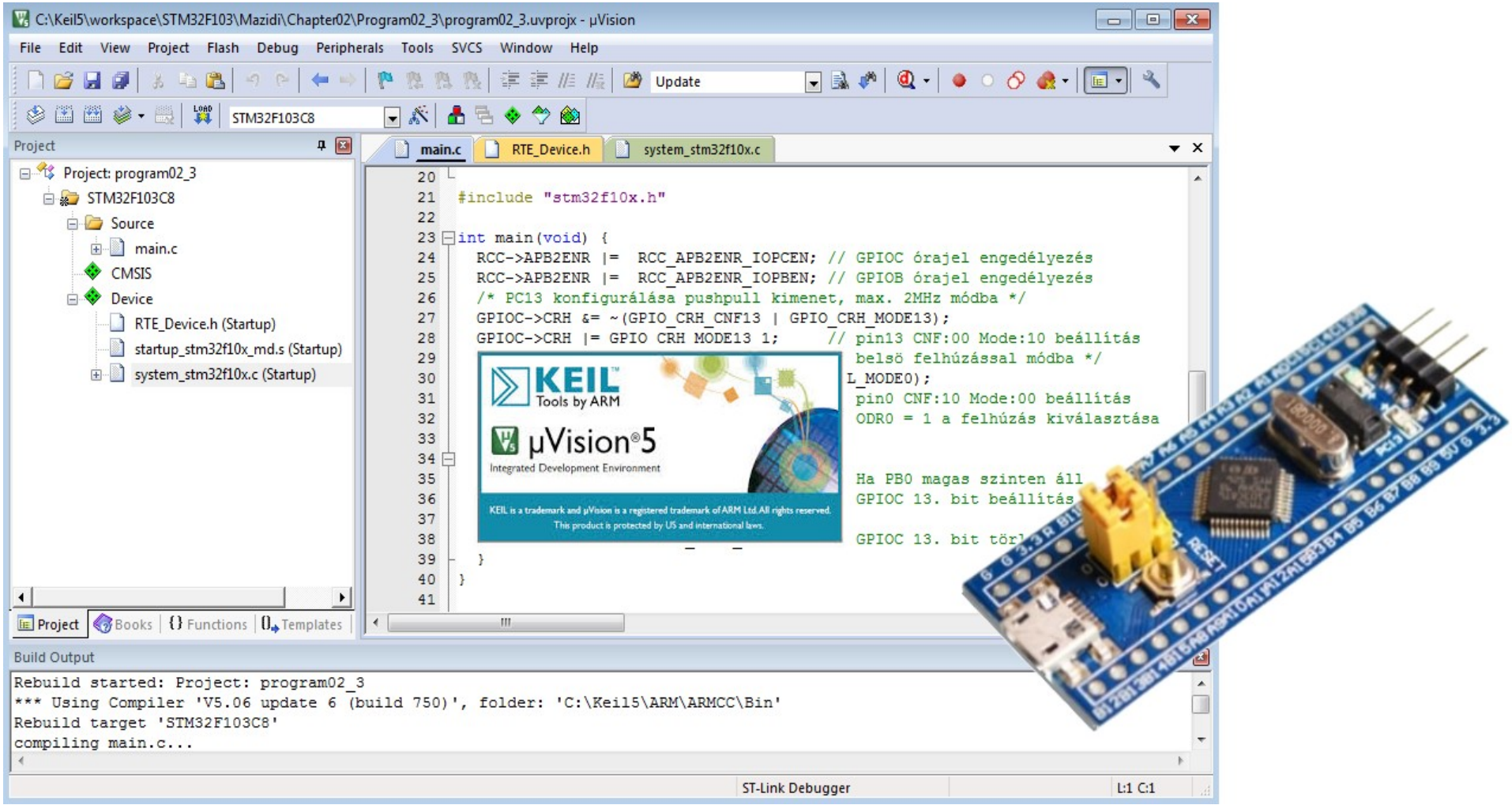
STM32 mikrovezérlők programozása ARM Keil környezetben



8. Analóg Digitális átalakító (ADC) – 2. rész

Felhasznált és ajánlott irodalom

- Joseph Yiu: Cortex-M for Beginners
- Joseph Yiu: The Definitive Guide To The ARM CORTEX-M3
- Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi: STM32 Arm Programming for Embedded Systems
- Alexander Tarasov: **Курс «Штурмуем STM32»**
- Warren Gay: Beginning STM32 - Developing with FreeRTOS, libopencm3 and GCC
- ARM Keil MDK Getting started
- **STM32F103C8** adatlap és termékinfo
- **STM32F103** Family Reference Manual

Az ADC-k üzemmódjainak áttekintése

- **Független módok:**
  - Egy csatorna, szingli konverzió
  - Pásztázás (több csatorna), szingli konverzió
  - Egy csatorna, folyamatos mód
  - Pásztázás, folyamatos mód
  - Injektált csatornák
- **Duál módok:**
  - Duál, reguláris, szimultán mód
  - Duál, gyors átlapolásos mód
  - Duál, lassú átlapolásos mód
  - Duál alternáló triggerelésű mód
  - Duál kombinált: reguláris/injektált szimultán mód
  - Duál kombinált: injektált szimultán + átlapolásos mód

Ajánlott olvasmány:

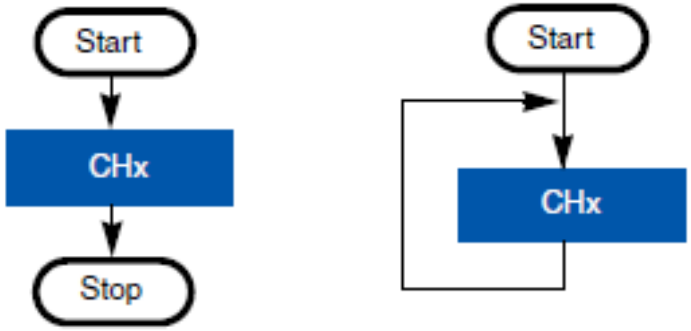
AN3116 Application note
STM32's ADC modes and their applications
(az **st.com** oldalon az AN3116 keresőszót írjuk be a megtalálásához!)

AN3116 mintaprogramok:
STSW-STM32028

Független módok

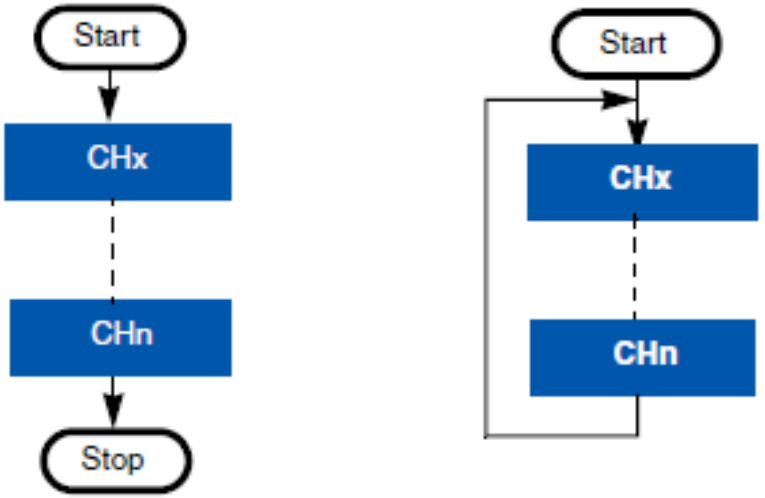
Egy csatorna

szingli konverzió folyamatos konverzió



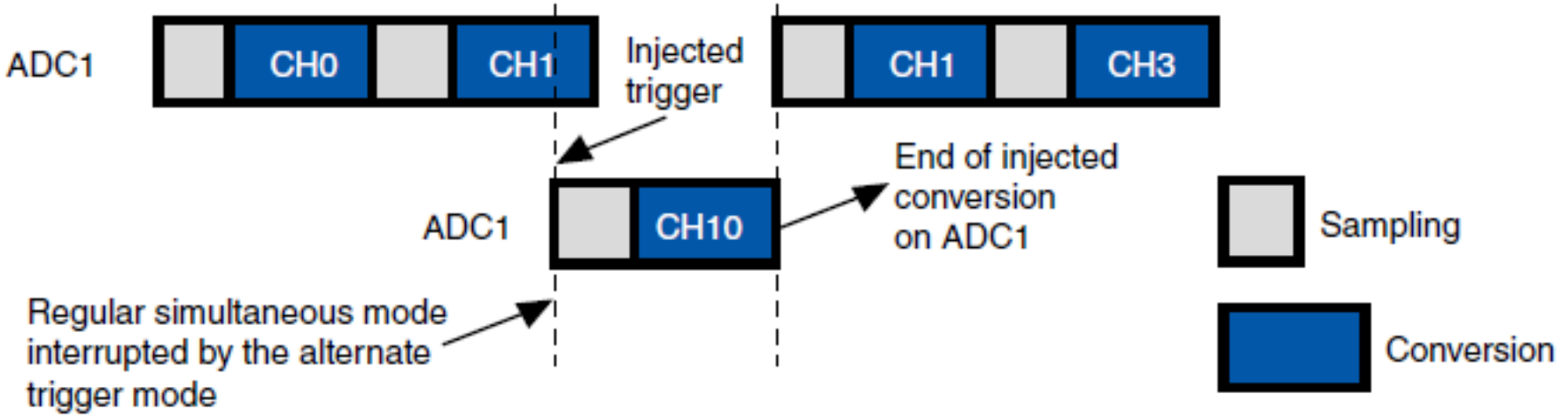
Pásztázás (több csatorna) (csak DMA-val lehetséges)

szingli konverzió folyamatos konverzió



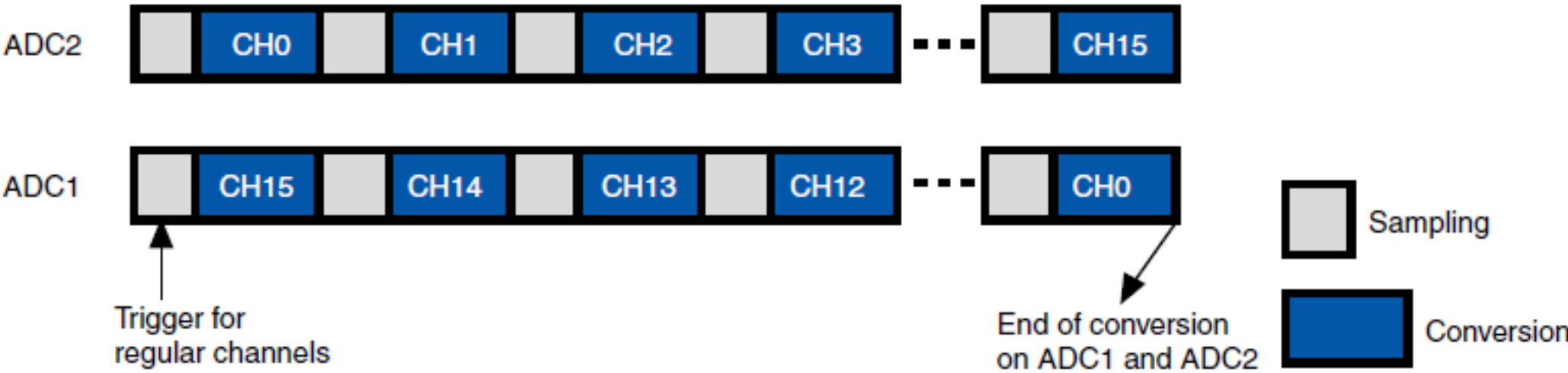
Injektált konverziós mód

(legfeljebb 4 csatorna injektálható)



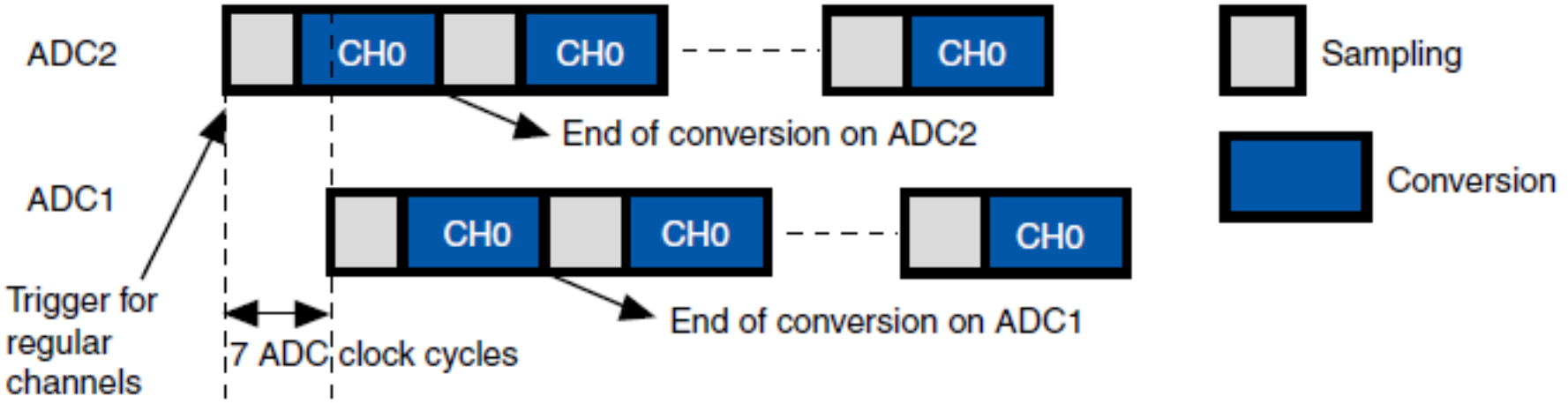
Duál, reguláris, szimultán mód

- Ebben a módban **ADC2** és **ADC1** egyszerre mintavételeznek és konvertálnak egy-egy csatornát (**ADC_CR1** DUALMOD=0110)
- Ez a mód akkor hasznos, amikor két, összetartozó jel egyidejű mérésére van szükség, mint pl. pillanatnyi feszültség és áram
- Duál módban **ADC2** alárendelt (slave) az eredmény a 32 bites **ADC1_DR** regiszterbe kerül (felső 16 bit az **ADC2** eredménye)
- Egyidejűleg ne mintavételezzük ugyanazon analóg csatorna jelét két ADC-vel, mert ez hibás eredményhez vezethet!



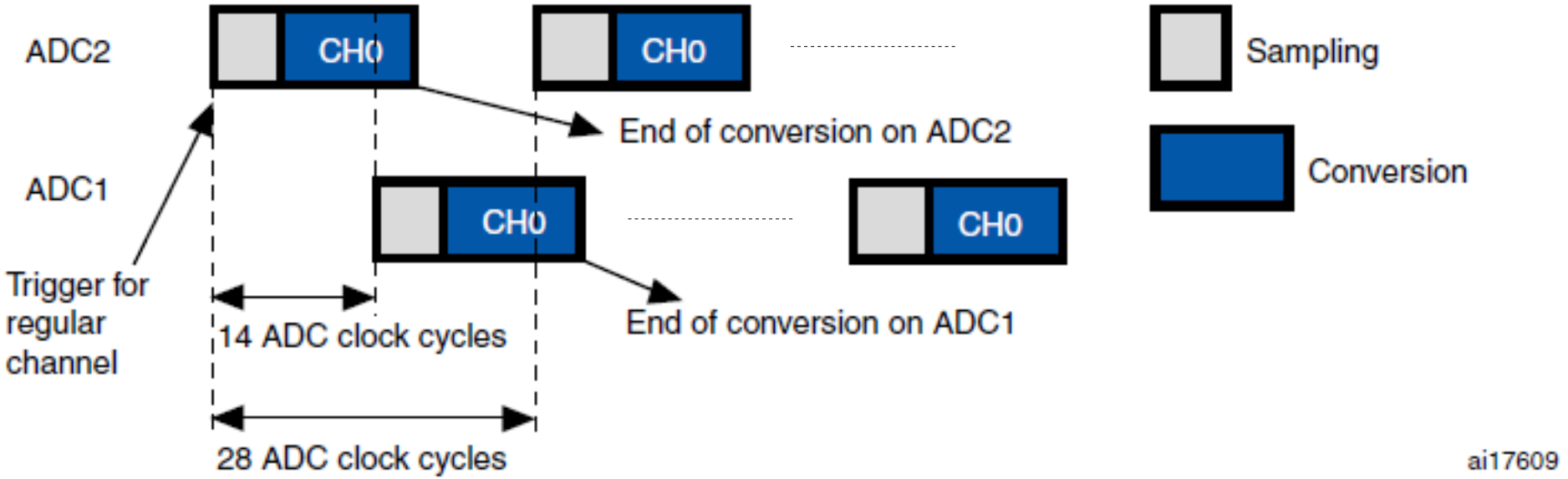
Duál, gyors átlapolásos mód

- A gyors átlapolás azt jelenti, hogy kis késleltetéssel, 7 ADC órajel ciklussal eltolva, felváltva történik a konverzió a két ADC-vel
- Az első konverzió **ADC2**-vel indul, majd 7 ciklus múlva **ADC1**
- A konverzió 12,5 ciklus, így mintavételezésre csak 1,5 ciklus marad, ezt kell beállítani konfigurálásnál
- Az adatok itt is a 32 bites **ADC1_DR** regiszterbe kerülnek



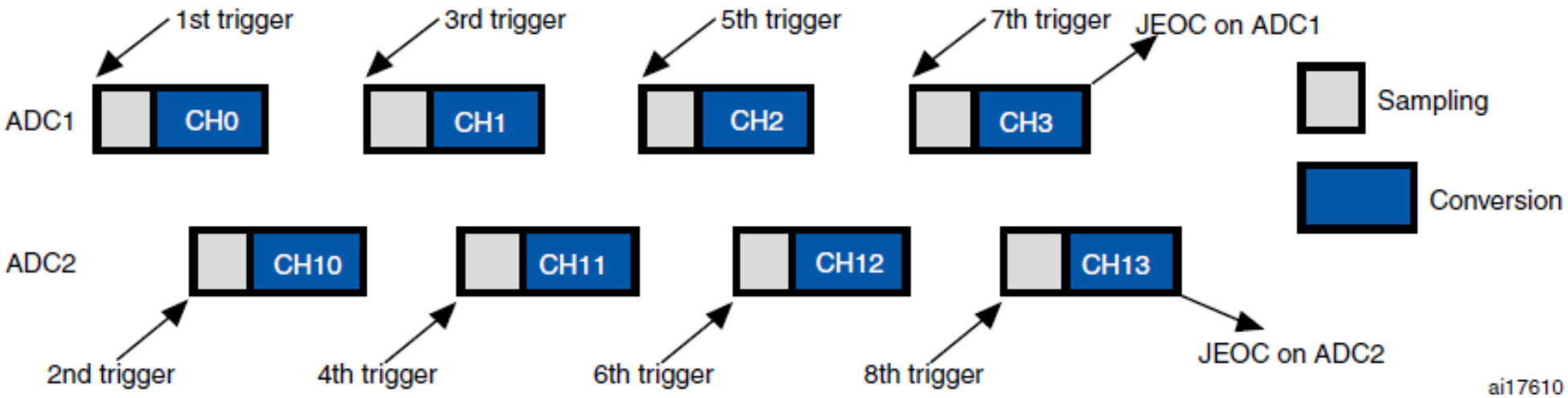
Duál, lassú átlapolásos mód

- A lassú átlapolás azt jelenti, hogy 14 ADC órajel késleltetéssel eltolva, felváltva történik a konverzió a két ADC-vel
- Az első konverzió **ADC2**-vel indul, majd 14 ciklus múlva **ADC1**
- Ebben az üzemmódban 1.5, 7.5, vagy 13.5 ciklus időtartamú mintavételezést használhatunk
- Az adatok itt is a 32 bites **ADC1_DR** regiszterbe kerülnek



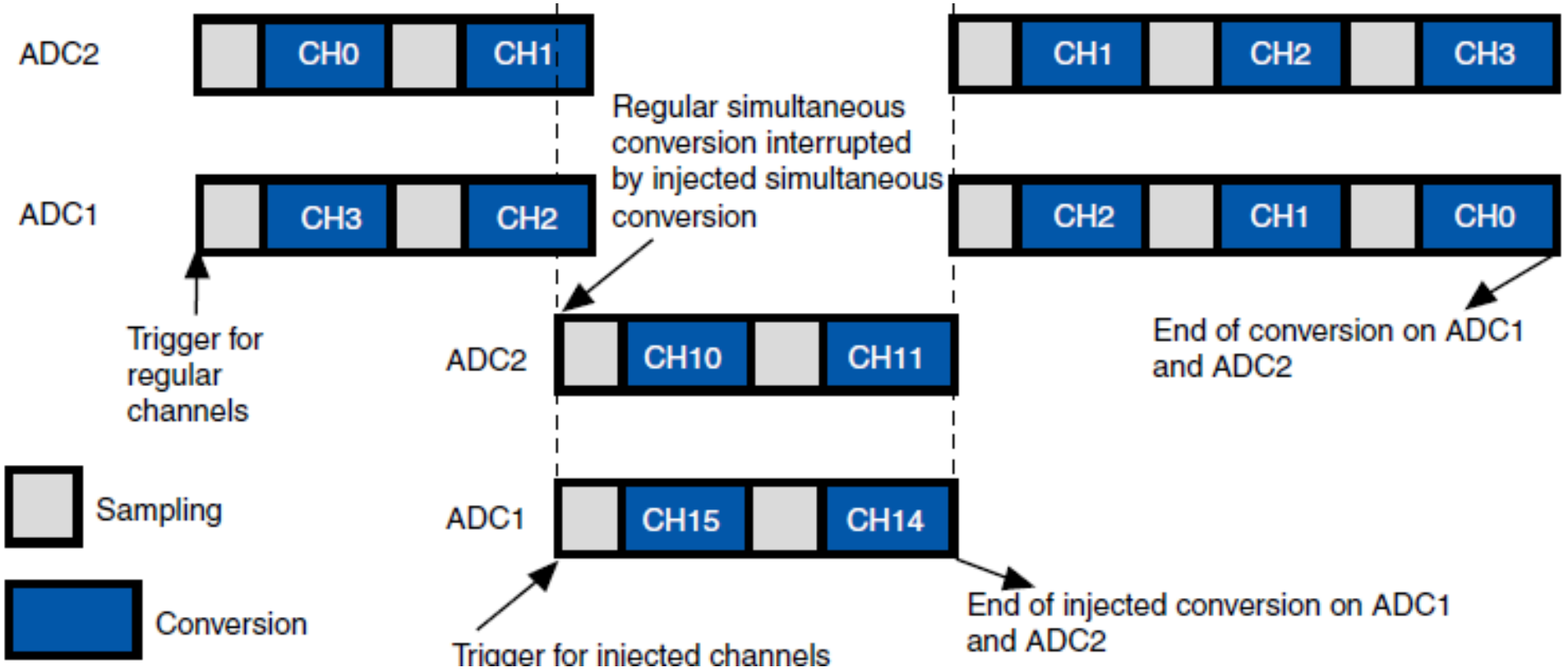
Duál, alternáló triggerelésű mód

- Ez a mód csak injektált csatornák pásztázására használható
- Az első triggerjel **ADC1**-et indítja, a következő **ADC2**-t és így tovább
- **Discontinuous** módban minden triggerjel csak a soron következő csatorna konverzióját indítja (ha **JDISCEN=1** az **ADC_CR1**-ben)
- **JDISCEN=0** esetén egy triggerjelre az összes injektált csatorna konverziója lefut és csak a végén billen be a konverzió végét jelző **JEOC** bit



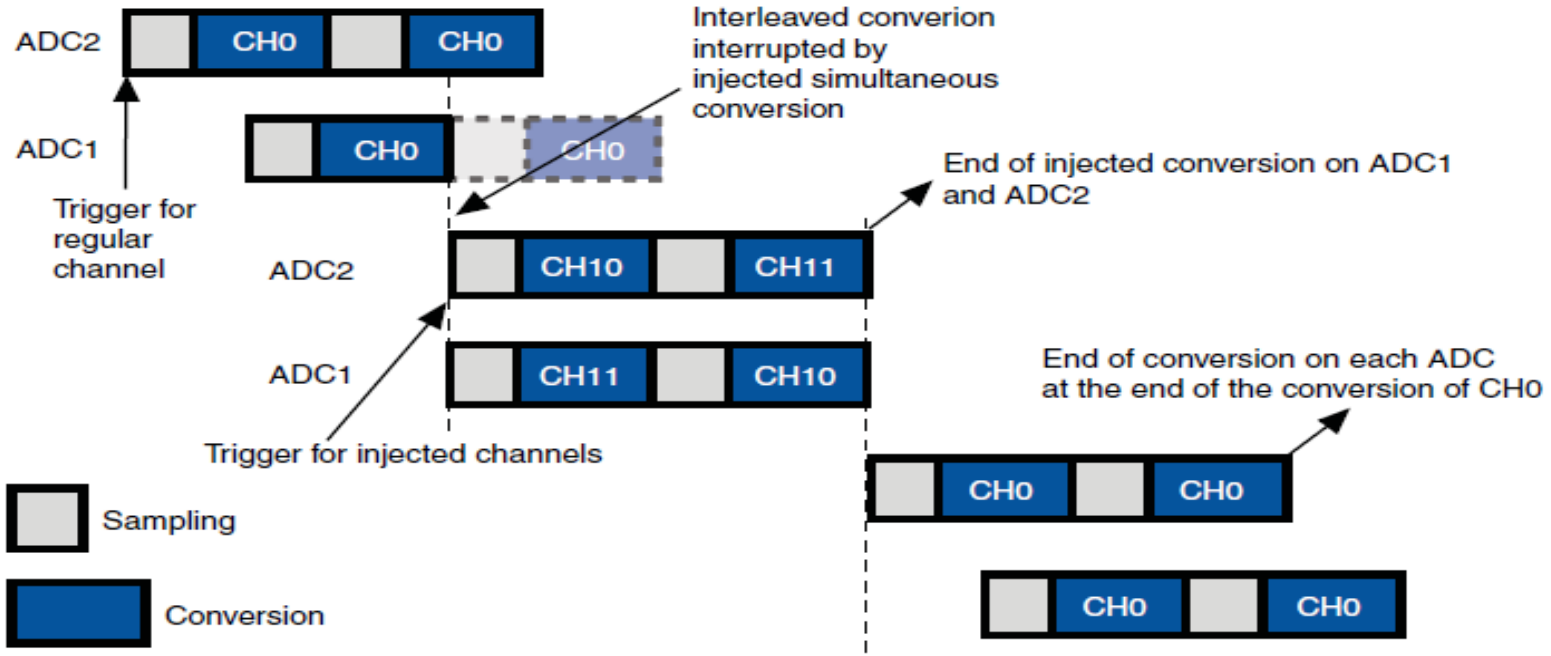
Duál reguláris + injektált szimultán mód

- Az ADC-k duális, szimultán módja csatornák injektálásával is kombinálható, ekkor az injektált csatornák konverziója is szimultán történik
- A szekvenciáknak azonos hosszúságúaknak kell lennie, vagy a triggerjel ne ismétlődjön a hosszabb szekvencia vége előtt!



Duál injektált szimultán + átlapolásos mód

- Ebben a módban a reguláris csatornák konverziója átlapolásos módban történik, az injektált csatornák konverziója pedig szimultán
- Példa: szünetmentes tápegységben az analóg watchdog a reguláris csatornákban figyeli a feszültséget, az injektált csatornákban pedig a terhelésen egyidejűleg mérjük a feszültséget és az áramot



Az ADC-k regiszterei

- Reguláris csatornák használatához

Regiszternév	Funkció
ADC_SR	ADC Status register
ADC_CR1	ADC Control Register 1.
ADC_CR2	ADC Control Register 2.
ADC_SMPR1	ADC sample time register 1.
ADC_SMPR2	ADC sample time register 2.
ADC_SQR1	ADC regular sequence register 1.
ADC_SQR2	ADC regular sequence register 2.
ADC_SQR3	ADC regular sequence register 3.
ADC_DR	ADC regular data register

- Injektált csatornák használatához

Regiszternév	Funkció
ADC_JSQR	ADC injected sequence register
ADC_JDR1	ADC injected data register 1.
ADC_JDR2	ADC injected data register 2.
ADC_JDR3	ADC injected data register 3.
ADC_JDR4	ADC injected data register 4.

Az ADC fontosabb regiszterei

- **ADC_SR** – ADC Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

STRT - Regular channel Start flag,
EOC – End of conversion (regular),
AWD – Analog watchdog flag

JSTRT – Injected channel Start flag
JEOC – End of conversion (injected),

- **ADC_CR1** – ADC Control Register 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SCAN – többcsatornás pásztázás engedélyezés, **JAUTO** – auto injektálás
DUALMODE – 0: független mód, 1 – 9: Duál mód (csak ADC1-ben állítható!)

Az ADC fontosabb regiszterei

- **ADC_CR2** – ADC Control register 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res.
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw	Res.		rw					rw	rw	rw	rw

EXTSEL[2:0] – triggerforrás választás
EXTTRIG – triggerelés engedélyezés
SWSTART – szoftveres triggerelés
ALIGN – jobbra/balra igazítás
DMA – DMA mód engedélyezés
CONT – folyamatos mód
ADON – ADC bekapcsolás
CAL – kalibráció indítása
JEXTSEL[2:0] – injektáló triggerforrás választás
JEXTTRIG – injektáló trigger engedélyezés

000: Timer 1 CC1 event
001: Timer 1 CC2 event
010: Timer 1 CC3 event
011: Timer 2 CC2 event
100: Timer 3 TRGO event
101: Timer 4 CC4 event
110: EXTI line 11
111: SWSTART

000: Timer 1 TRGO event
001: Timer 1 CC4 event
010: Timer 2 TRGO event
011: Timer 2 CC1 event
100: Timer 3 CC4 event
101: Timer 4 TRGO event
110: EXTI line15/TIM8_CC4 event
111: JSWSTART

Az ADC fontosabb regiszterei

- **ADC_SMPR1/ADC_SMPR2** – ADC mintavételezési idő megadása

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
Res.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

000: 1.5 cycles
001: 7.5 cycles
010: 13.5 cycles
011: 28.5 cycles
100: 41.5 cycles
101: 55.5 cycles
110: 71.5 cycles
111: 239.5 cycles

A mintavételezési idő csatornánként megadható
Megjegyzés: a sorszám itt nem a mérési sorrend, hanem az analóg bemenet sorszáma (AN0 - AN17)

Az ADC fontosabb regiszterei

- **ADC_SQR1** – ADC szekvenciák megadása

darabszám – 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]						SQ14[4:0]					SQ13[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- **ADC_SQR2**

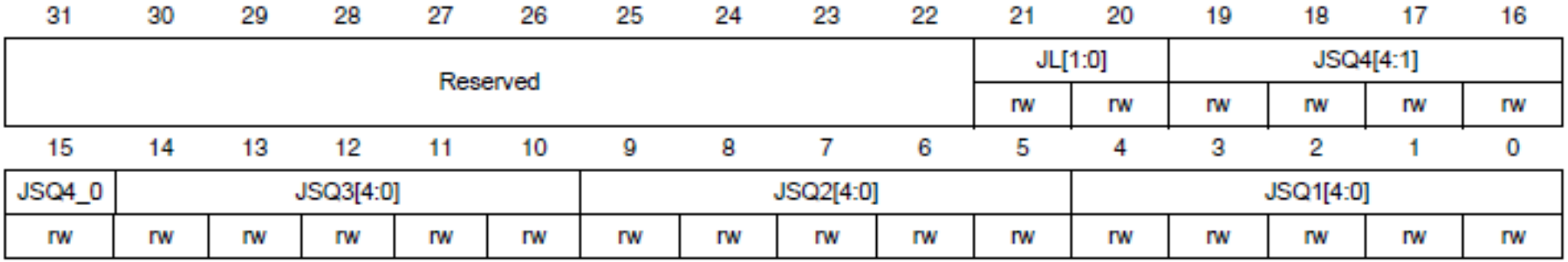
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]						SQ8[4:0]					SQ7[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- **ADC_SQR3**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]						SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Az ADC fontosabb regiszterei

- **ADC_JSQR** – injektált szekvencia megadása



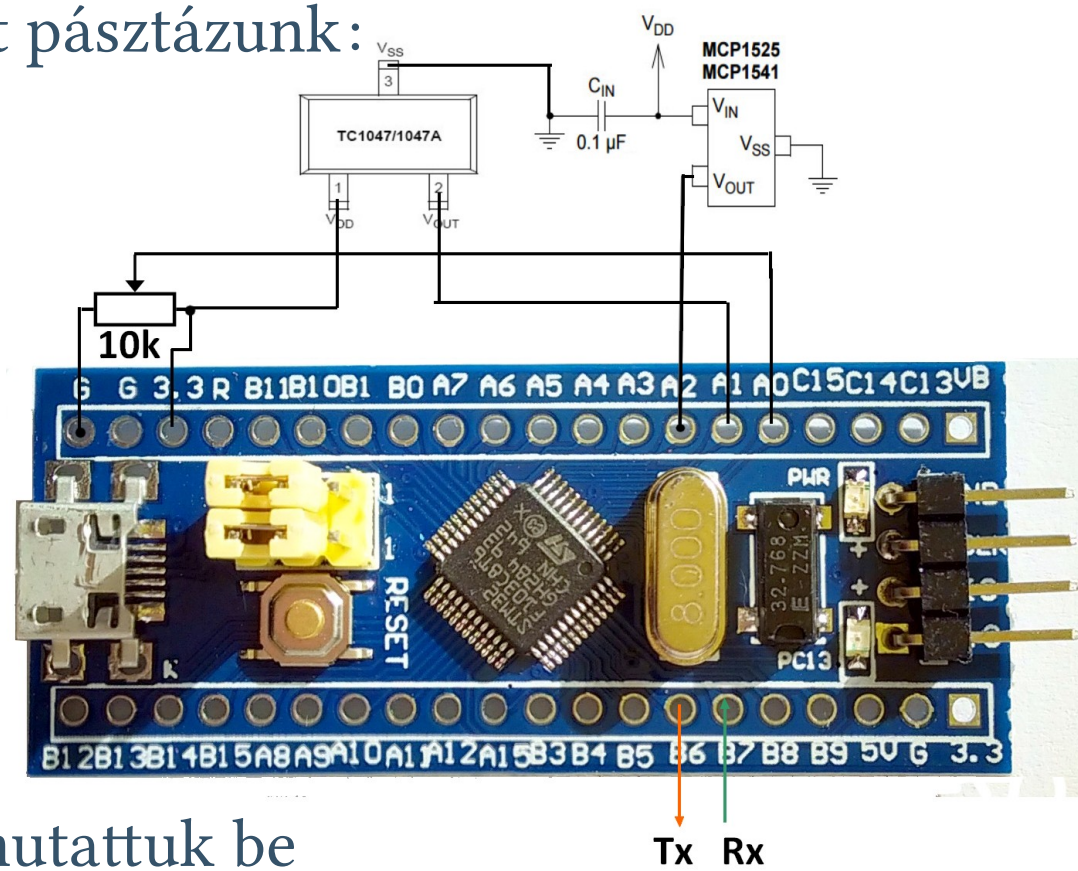
- **JL[1:0]** – az injektált szekvencia hossza (**00:** 1, **01:** 2, **10:** 3, **11:** 4)
- **JSQ*n*[4:0]** – az *n*-edik injektált csatorna sorszáma (0..17)

- Például 4 db injektált csatorna (AN16, AN17, AN2, AN3) esetén:

```
ADC1->JSQR = (16<<0) |          // AN16: a belső hőmérő
             (17<<5) |          // AN17: a belső referencia
             (2<<10) |          // AN2:  PA2 külső jel
             (3<<15) |          // AN3:  PA3 külső jel
             (3<<20);           // A konverzió sorozat hossza = 4
```

Program07_3: reguláris csatornapásztázás

- Ha több reguláris csatornát akarunk pásztázni, akkor 1-be kell állítani **ADC_CR1**-ben a **SCAN** bitet
- Scan módban használnunk kell a DMA adatátvitelt: 1-be állítjuk **ADC_CR2** DMA bitjét és konfiguráljuk **DMA1** CH1 csatornáját
- A mintaprogramban 5 csatornát pásztázunk:
  AN0: potméter
  AN1: TC1047 hőmérő
  AN2: MCP1525 Vref
  AN16: belső hőmérő
  AN17: belső referencia
- A mérést **TIM2CH2** triggereli
- Azt eredményeket **UART1**-en (**PB6**) íratjuk ki
- **UART1** kezelését a 6. előadás **uart_irq** mintaprogramjában mutattuk be



Program07_3/main.c

```c
#include "stm32f10x.h"
#include <stdio.h>
extern void buffer_Init (void);
extern void USART1_init (void);

int main (void) {
  uint16_t data[6] = {0,0,0,0,0,0};
  double volt, temp;
  buffer_Init();                          // RX / TX bufferek inicializálása
  USART1_init();                          // USART1 configuration
    GPIO_Init();
  Timer2_Init();                          // TIM2 konfigurálás: 1 Hz trigger
  ADC1_Init();                            // ADC1 konfigurálás belsö homerohöz
  DMA1_init();
  __enable_irq();                         // Megszakítások globális engedélyezése
  printf("ADC multichannel scan \r\n");
  while (1) {                             // végtelen ciklus
    DMA1_ch1_setup((unsigned int)&ADC1->DR, (unsigned int)data, 5);
    while(!(DMA1->ISR & DMA_ISR_GIF1));   // Konverzió végére várunk
    DMA1->IFCR |= DMA_IFCR_CGIF1;         // Törli az 1. csatorna jelzobitjeit
    ADC1->SR = 0;                         // Clear flags
    printf("%d, %d, %d, %d, %d\r\n", data[0], data[1], data[2], data[3], data[4]);
  }
}
```

Program07_3/main.c

```
/*-----------------------------------------------
  PA0, PA1, PA2,PA3 analóg bemenetek konfigurálása
  -----------------------------------------------*/
void GPIO_Init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;     // GPIOA engedélyezés
  GPIOA->CRL &= 0xFFFFF000;               // PA0,1,2 analóg bemenet
}

/*-----------------------------------------------
  TIMER2 CH2 konfigurálása 1 Hz-es triggereléshez
  -----------------------------------------------*/
void Timer2_Init() {
  RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;     // TIM2 órajel engedélyezése
  TIM2->CR1 = 0;
  TIM2->CR2 = 0;
  TIM2->PSC = SystemCoreClock/10000 - 1; // 10 kHz-re osztjuk le az órajelet
  TIM2->ARR = 10000 - 1;                  // 1000-ig számlálunk (1000 ms)
  TIM2->CNT = 0;                          // Számláló nullázása
  TIM2->CCMR1 = 0x6800;                   // Ch2 PWM1 mód, preload enable
  TIM2->CCER = TIM_CCER_CC2E;             // Ch2 engedélyezése
  TIM2->CCR2 = 50-1;
  TIM2->CR1 |= TIM_CR1_CEN;               // Számlálás engedélyezés
}
```

Program07_3/main.c

```
/*-------------------------------------------------------------
  ADC1 konfigurálása belsö höméröhöz és TIM2CH2 triggereléshez
  -------------------------------------------------------------*/
void ADC1_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;     // ADC1 órajel engedélyezés
  RCC->CFGR &= ~RCC_CFGR_ADCPRE_0;        // ADC prescaler = 6
  RCC->CFGR |=  RCC_CFGR_ADCPRE_1;        // ADCCLK = 72/6 = 12 MHz
  ADC1->CR1 = ADC_CR1_SCAN;               // Pásztázás engedélyezése
  ADC1->CR2 = ADC_CR2_EXTTRIG |           // External trigger engedélyezés
              ADC_CR2_EXTSEL_0 |          // EXTSEL=011, TIM2 CC2 esemény
              ADC_CR2_EXTSEL_1 |
              ADC_CR2_DMA;                // DMA mód engedélyezése
  ADC1->SMPR1 = ADC_SMPR1_SMP16 |         // Mintavétel: 111, azaz 239.5 ciklus
                ADC_SMPR1_SMP17;
  ADC1->SMPR2 = ADC_SMPR2_SMP0 |
                ADC_SMPR2_SMP1 |
                ADC_SMPR2_SMP2 ;
  ADC1->SQR3 = (0)      |                 // AN0: potméter jele
               (1<<5)   |                 // AN1: TC1047 hőmérő
               (2<<10)  |                 // AN2: MCP1525 Vref
               (16<<15) |                 // AN16: a belső hőmérő
               (17<<20);                  // AN17: a belső referencia
  ADC1->SQR1 = (4<<20);                   // A konverzió sorozat hossza=5
  ADC1->CR2 |= ADC_CR2_ADON |             // ADC1 engedélyezése
               ADC_CR2_TSVREFE;           // A belső referencia bekapcsolása
}
```

Program07_3/main.c

```
/*-------------------------------------------------------------------------
   A DMA1 vezérlő 1. csatorna inicializálása
 *-------------------------------------------------------------------------*/
void DMA1_init(void) {
  RCC->AHBENR |= RCC_AHBENR_DMA1EN;          // DMA vezérlo engedélyezése
  DMA1->IFCR |= DMA_IFCR_CGIF1;              // Törli 1. csatorna jelzöbitjeit
}

/*-------------------------------------------------------------------------
   DMA 1. csatorna <-- ADC adatátvitel konfigurása és indítása
 *-------------------------------------------------------------------------*/
void DMA1_ch1_setup(unsigned int paddr, unsigned int maddr, int len) {
  DMA1_Channel1->CCR = 0;                    // DMA1 Channel 4 ideiglenes letiltása
  while (DMA1_Channel1->CCR & 1) {}          // Vár, amíg DMA1 Channel 1 aktív
  DMA1->IFCR |= DMA_IFCR_CGIF1;              // Törli az 1. csatorna jelzőbitjeit
  DMA1_Channel1->CPAR = paddr;               // Destination  address  (cél)
  DMA1_Channel1->CMAR = maddr;               // Source address  (forrás)
  DMA1_Channel1->CNDTR = len;                // length (adatblokk hossza)
  DMA1_Channel1->CCR = DMA_CCR1_MSIZE_0 |    // Memory size = 16
                       DMA_CCR1_PSIZE_0 |    // Peripheral size = 16
                       DMA_CCR1_MINC    |    // Memory increment mode enable
                       DMA_CCR1_PL_1;        // Priority level = High
  // DMA_CCR1_DIR = 0                        // Read form peripherial
  // DMA_CCR1_PINC = 0                       // Peripheral increment mode disabled
  DMA1_Channel1->CCR |= DMA_CCR1_EN;         // DMA1 Channel 1 engedélyezése
}
```

Program07_4: injektált csatornák pásztázása

- Az injektált csatornák (max. 4 db) pásztázásához nem kell **DMA,** mert mindegyik csatornának saját adatregisztere van
- Az alábbi példában az **AN0, AN1, AN2, AN3** csatornákat pásztázzuk (a triggerelést **Timer2 CH1** csatornája végzi, 1 másodpercenként
- Az eredemények kiíratása most is **UART1**-en keresztül (**PB6** kivezetés) történik
- Az előző programhoz képest könnyebség, hogy nem kell **DMA**-t használni, nem kell adattömböt deklarálni, az adatokat közvetlenül az **ADC1_JDR1, ADC1_JDR2, ADC1_JDR3, ADC1_JDR4** regiszterekből vehetjük elő
- A pásztázás végét most az **ADC1_SR** regiszter **JEOC** bitjének 1-be billenése jelzi

Program07_4/main.c

```c
#include "stm32f10x.h"
#include <stdio.h>

extern void buffer_Init (void);
extern void USART1_init (void);
void Timer2_Init(void);
void ADC1_Init(void);
void GPIO_Init(void);

/*-----------------------------------------------------------------------
  MAIN function
 *-----------------------------------------------------------------------*/
int main (void) {
  buffer_Init();                          // RX / TX bufferek inicializálása
  USART1_init();                          // USART1 configuration
  GPIO_Init();
  Timer2_Init();                          // TIM2 konfigurálás: 1 Hz trigger
  ADC1_Init();                            // ADC1 konfigurálás belsö homerohöz
  __enable_irq();                         // Megszakítások globális engedélyezése
  printf("ADC multichannel scan \r\n");
  while (1) {                             // végtelen ciklus
    while(!(ADC1->SR & ADC_SR_JEOC));     // Konverzió végére várunk
    ADC1->SR = 0;                         // Clear flags
    printf("%d, %d, %d, %d\r\n", ADC1->JDR1, ADC1->JDR2, ADC1->JDR3, ADC1->JDR4);
  }
}
```

Program07_4/main.c

```
/*-----------------------------------------------
  PA0, PA1, PA2,PA3 analóg bemenetek konfigurálása
  -----------------------------------------------*/
void GPIO_Init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;     // GPIOA engedélyezés
  GPIOA->CRL &= 0xFFFF0000;               // PA0,1,2,3 analóg bemenet
}

/*-----------------------------------------------
  TIMER2 CH1 konfigurálása 1 Hz-es triggereléshez
  -----------------------------------------------*/
void Timer2_Init() {
  RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;     // TIM2 órajel engedélyezése
  TIM2->CR1 = 0;
  TIM2->CR2 = 0;
  TIM2->PSC = 7199;                       // 10 kHz-re osztjuk le az órajelet
  TIM2->ARR = 4999;                       // 5000-ig számlálunk (1000 ms periódus)
  TIM2->CNT = 0;                          // Számláló nullázása
  TIM2->CCMR1 = 0x0038;                   // Ch1 toggle mód, preload enable
  TIM2->CCER = TIM_CCER_CC1E;             // Ch1 engedélyezése
  TIM2->CCR1 = 50-1;
  TIM2->CR1 |= TIM_CR1_CEN;               // Számlálás engedélyezés
}
```

Program07_4/main.c

```
/*-------------------------------------------------------------
  ADC1 konfigurálása injektált scan és TIM2CH1 triggereléshez
  -------------------------------------------------------------*/
void ADC1_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;     // ADC1 órajel engedélyezés
  RCC->CFGR &= ~RCC_CFGR_ADCPRE_0;        // ADC prescaler = 6
  RCC->CFGR |=  RCC_CFGR_ADCPRE_1;        // ADCCLK = 72/6 = 12 MHz
  ADC1->CR1 = ADC_CR1_SCAN;               // Pásztázás engedélyezése
  ADC1->CR2 = ADC_CR2_JEXTTRIG |          // Injektálás triggerelés engedélyezés
              ADC_CR2_JEXTSEL_0 |         // JEXTSEL=011, TIM2 CC1 event
              ADC_CR2_JEXTSEL_1;
  ADC1->SMPR2 = ADC_SMPR2_SMP0 |
                ADC_SMPR2_SMP1 |
                ADC_SMPR2_SMP2 |
                ADC_SMPR2_SMP3 ;
  ADC1->JSQR = (0)     |                  // AN0: potméter jele
               (1<<5)  |                  // AN1: TC1047 hömérö
               (2<<10) |                  // AN2: MCP1525 Vref
               (3<<15) |                  // AN3: a fényérzékeny osztó
               (3<<20);                   // A konverzió sorozat hossza=4
  ADC1->CR2 |= ADC_CR2_ADON;              // ADC1 engedélyezése
}
```

Program07_5: pásztázás, a legegyszerűbben

- Öt csatornában végzünk pásztázó mérést, **DMA** nélkül, szoftveres indítással, a reguláris (AN17) és a 4 injektált (AN0, AN1, AN2, AN3) csatorna mérése a fenti sorrendben történik
- Pásztázáshoz **ADC1_CR1**-ben most a **SCAN** és **JAUTO** biteket kell 1-be állítani (az injektálást így **EOC** triggereli)
- A szoftveres indítás konfigurálása ugyanúgy történik, mint a **Program07_1** mintapéldában (**ADC_CR2**-ben **EXTSEL=111, EXTTRIG=1**)
- Az **SWSTART** bit 1-be állítása a reguláris csatorna konverzióját indítja, az injektált szekvencia pedig utána automatikusan indul a **JAUTO=1** beállítás következtében
- A pásztázás végét az **ADC1_SR** regiszter **JEOC** bitje jelzi

Program07_5/main.c

```
#include "stm32f10x.h"
#include <stdio.h>
extern void buffer_Init (void);
extern void USART1_init (void);
void ADC1_Init(void);
void GPIO_Init(void);
void delayMs(int n);
int main (void) {
  buffer_Init();                          // RX / TX bufferek inicializálása
  USART1_init();                          // USART1 configuration
  GPIO_Init();
  ADC1_Init();                            // ADC1 konfigurálás belsö homerohöz
  __enable_irq();                         // Megszakítások globális engedélyezése
  printf("ADC multichannel scan \r\n");
  while (1) {                             // végtelen ciklus
    ADC1->CR2 |= ADC_CR2_SWSTART;         // Konverzió indítása
    while(!(ADC1->SR & ADC_SR_JEOC));     // Konverzió végére várunk
    ADC1->SR = 0;                         // Clear flags
    printf("%d, %d, %d, %d, %d\r\n",ADC1->DR,ADC1->JDR1,ADC1->JDR2,ADC1->JDR3,ADC1->JDR4);
    delayMs(1000);
  }
}

void GPIO_Init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;     // GPIOA engedélyezés
  GPIOA->CRL &= 0xFFFF0000;               // PA0,1,2,3 analóg bemenet
}
```

Program07_5/main.c

```
/*-------------------------------------------------------------
  ADC1 konfigurálása 1 reguláris + 4 injektált scan méréshez
  -------------------------------------------------------------*/
void ADC1_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;     // ADC1 órajel engedélyezés
  RCC->CFGR &= ~RCC_CFGR_ADCPRE_0;        // ADC prescaler = 6
  RCC->CFGR |=  RCC_CFGR_ADCPRE_1;        // ADCCLK = 72/6 = 12 MHz
  ADC1->CR1 = ADC_CR1_SCAN |              // Pásztázás engedélyezése
              ADC_CR1_JAUTO;              // Automatikus injektálás
  ADC1->CR2 = ADC_CR2_EXTTRIG |           // Triggerelés engedélyezés
              ADC_CR2_EXTSEL;             // EXTSEL=111, SWSTART
  ADC1->SMPR1 = ADC_SMPR1_SMP17;          // AN17 mintavételezés (111)
  ADC1->SMPR2 = ADC_SMPR2_SMP0 |          // AN0..AN3 mintavételezési idő
                ADC_SMPR2_SMP1 |          // maximálisra állítása (111)
                ADC_SMPR2_SMP2 |
                ADC_SMPR2_SMP3 ;
  ADC1->SQR1 = (0<<20);                   // Reguláris csatornák száma-1
  ADC1->SQR3 = 17;                        // Belső referencia
  ADC1->JSQR = (0)     |                  // AN0: potméter jele
               (1<<5)  |                  // AN1: TC1047 hőmérő
               (2<<10) |                  // AN2: MCP1525 Vref
               (3<<15) |                  // AN3: a fényérzékeny osztó
               (3<<20);                   // A konverzió sorozat hossza=4
  ADC1->CR2 |= ADC_CR2_TSVREFE |          // Belső referencia bekapcsolása
               ADC_CR2_ADON;              // ADC1 engedélyezése
}
```

Program07_6: duál mód, gyors átlapolással

- A gyors átlapolásos duál mód segítségével megduplázhatjuk a mintavételezési sebességet. Pl. 14 MHz-es ADC frekvencia esetén a 7 ciklusonként indított konverzió 2 MHz-es mintavételezést jelent
- A duál üzemmódokat az **ADC_CR1** regiszter **DUALMODE[3:0]** bitjeiben konfigurálhatjuk:

0000: Independent mode.
0001: Combined regular simultaneous + injected simultaneous mode
0010: Combined regular simultaneous + alternate trigger mode
0011: Combined injected simultaneous + fast interleaved mode
0100: Combined injected simultaneous + slow Interleaved mode
0101: Injected simultaneous mode only
0110: Regular simultaneous mode only
0111: Fast interleaved mode only
1000: Slow interleaved mode only
1001: Alternate trigger mode only

- Ebben a programban mindkét **AN0** csatorna jelét mérjük, folyamatos módban, s DMA-val pakoljuk egy tömbbe

Program07_6/main.c

```
#include "stm32f10x.h"
#include <stdio.h>
void GPIO_Init(void);
void ADC1_Init(void);
void ADC2_Init(void);
void DMA1_init(void);
void DMA1_ch1_setup(void);
#define BLOCK_SIZE   256
int DATA_BLOCK[BLOCK_SIZE];
int main (void) {
  GPIO_Init();                              // PA0 (AN0) konfigurálása
  ADC1_Init();                              // ADC1 konfigurálás belső hőmérőhöz
  ADC2_Init();                              // ADC1 konfigurálás belső hőmérőhöz
  DMA1_init();
  __enable_irq();                           // Megszakítások globális engedélyezése
  while (1) {                               // végtelen ciklus
    DMA1_ch1_setup();
    ADC1->CR2 |= ADC_CR2_SWSTART;           // Konverzió indítása
    while(!(DMA1->ISR & DMA_ISR_TCIF1)); // Konverzió végére várunk
    DMA1->IFCR |= DMA_IFCR_CGIF1;           // Törli az 1. csatorna jelzőbitjeit
    //--- Itt kellene valamit kezdeni az adatokkal -----
  }
}
```

Program07_6/main.c

```
/*-------------------------------------------------------------
  ADC1 konfigurálása duál, gyors átfedéses, folyamatos módhoz
  -------------------------------------------------------------*/
void ADC1_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;     // ADC1 órajel engedélyezés
  RCC->CFGR &= ~RCC_CFGR_ADCPRE_0;        // ADC prescaler = 6
  RCC->CFGR |=  RCC_CFGR_ADCPRE_1;        // ADCCLK = 72/6 = 12 MHz
  ADC1->CR2 &= ~ADC_CR2_ADON;             // ADC1 lekapcsolása
  ADC1->CR1 = ADC_CR1_DUALMOD_0 |         // 0011: Fast interleaved mód
              ADC_CR1_DUALMOD_1;
  ADC1->CR2 = ADC_CR2_EXTTRIG |           // External trigger engedélyezés
              ADC_CR2_EXTSEL |            // EXTSEL=111: software start
              ADC_CR2_DMA;                // DMA mód engedélyezése
  ADC1->SMPR1 = 0;                        // Mintavétel 1.5 ciklus
  ADC1->SQR3 = 0;                         // AN0: potméter jele
  ADC1->SQR1 = (0<<20);                   // A konverzió sorozat hossza=1
  ADC1->CR2 |= ADC_CR2_CONT |             // CONT=1 folyamatos mód
               ADC_CR2_ADON;              // ADC1 engedélyezése
}
```

Program07_6/main.c

```
/*-------------------------------------------------------------
  ADC2 konfigurálása duál, gyors átfedéses, folyamatos módhoz
  -------------------------------------------------------------*/
void ADC2_Init() {
  RCC->APB2ENR |= RCC_APB2ENR_ADC2EN;     // ADC2 órajel engedélyezés
  ADC2->CR2 &= ~ADC_CR2_ADON;             // ADC2 lekapcsolása
  ADC2->CR1 = ADC_CR1_DUALMOD_0 |         // 0011: Fast interleaved mód
              ADC_CR1_DUALMOD_1;
  ADC2->CR2 = ADC_CR2_EXTTRIG |           // External trigger engedélyezés
              ADC_CR2_EXTSEL |            // EXTSEL=111: software start
              ADC_CR2_DMA;                // DMA mód engedélyezése
  ADC2->SMPR1 = 0;                        // Mintavétel 1.5 ciklus
  ADC2->SQR3 = 0;                         // AN0: potméter jele
  ADC2->SQR1 = (0<<20);                   // A konverzió sorozat hossza=1
  ADC2->CR2 |= ADC_CR2_CONT |             // CONT=1 folyamatos mód
               ADC_CR2_ADON;              // ADC2 engedélyezése
}

/*-----------------------------------------------
  PA0 konfigurálása analóg bemenetnek
  -----------------------------------------------*/
void GPIO_Init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;     // GPIOA engedélyezés
  GPIOA->CRL &= 0xFFFFFFF0;               // PA0 (AN0) analóg bemenet
}
```

Ezek itt hatástalanok (ADC1 dominál)

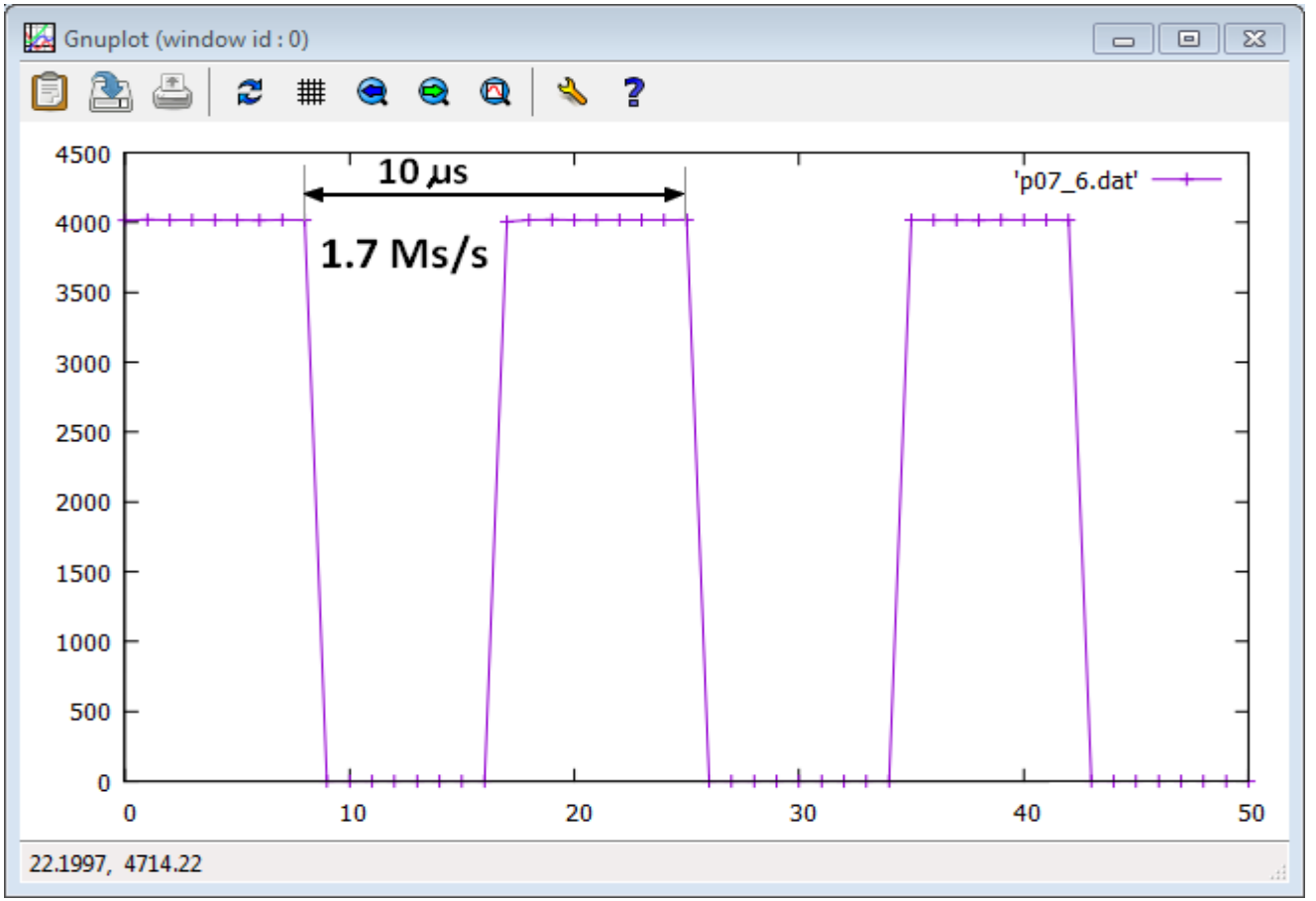
Program07_6/main.c

```
/*-------------------------------------------------------------------------
   A DMA1 vezérlő 1. csatorna inicializálása
 *-------------------------------------------------------------------------*/
void DMA1_init(void) {
  RCC->AHBENR |= RCC_AHBENR_DMA1EN;          // DMA vezérlő engedélyezése
  DMA1->IFCR |= DMA_IFCR_CGIF1;              // Törli az 1. csatorna jelzőbitjeit
}

/*-------------------------------------------------------------------------
   DMA 1. csatorna <-- ADC adatátvitel konfigurása és indítása
 *-------------------------------------------------------------------------*/
void DMA1_ch1_setup(void) {
  DMA1_Channel1->CCR = 0;                    // DMA1 Channel 4 ideiglenes letiltása
  while (DMA1_Channel1->CCR & 1) {}          // Vár, amíg DMA1 Channel 1 aktív
  DMA1->IFCR |= DMA_IFCR_CGIF1;              // Törli az 1. csatorna jelzőbitjeit
  DMA1_Channel1->CPAR = (int)&ADC1->DR;      // Source  address  (forráscím)
  DMA1_Channel1->CMAR = (int)DATA_BLOCK;     // Destination address  (cél)
  DMA1_Channel1->CNDTR = BLOCK_SIZE;         // length (adatblokk hossza)
  DMA1_Channel1->CCR = DMA_CCR1_MSIZE_1 |    // Memory size = 32
                       DMA_CCR1_PSIZE_1 |    // Peripheral size = 32
                       DMA_CCR1_MINC    |    // Memory increment mode enable
                       DMA_CCR1_PL_1;        // Priority level = High
  // DMA_CCR1_DIR = 0                        // Read form peripherial
  // DMA_CCR1_PINC = 0                       // Peripheral increment mode disabled
  DMA1_Channel1->CCR |= DMA_CCR1_EN;         // DMA1 Channel 1 engedélyezése
}
```

Program07_6b: 100 kHz jel vizsgálata

- **Program07_6** kiegészítve egyszerű kiíratással és **TIM3CH3** segítségével 100 kHz-es négyszögjelet generálunk a **PB0** kimeneten
- A kiírt adatokat az Arduino IDE **Serial Plotter** ablakában megjeleníthetjük, vagy fájlba írathatjuk
- 12 MHz-es ADC órajel esetén kb. 1,7 Ms/s mintavételi sebesség érhető el



Program07_6b/main.c – a változtatások

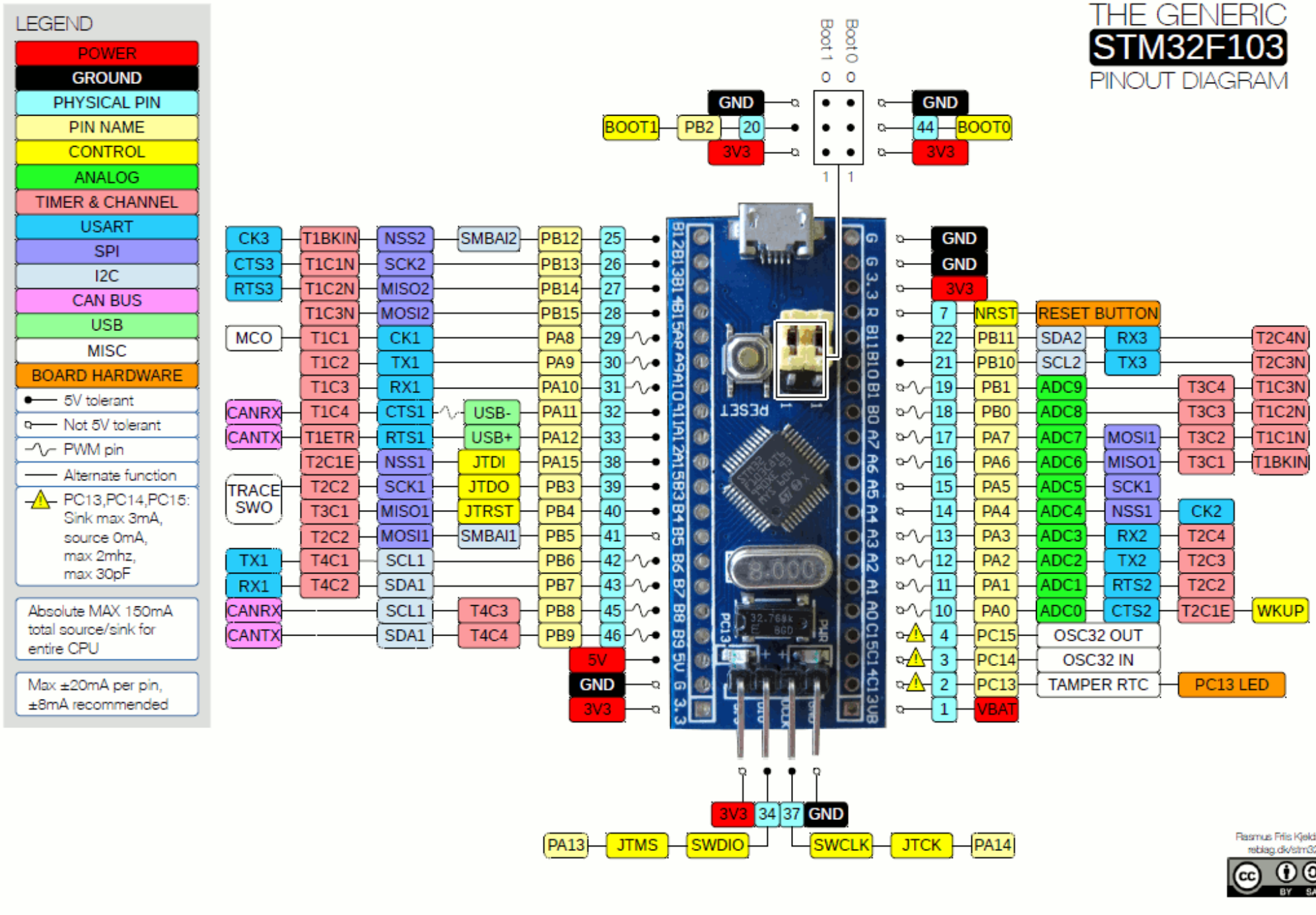
```
int main (void) {
  buffer_Init();                          // RX / TX bufferek inicializálása
  USART1_init();                          // USART1 configuration
  GPIO_Init();
  ADC1_Init();                            // ADC1 konfigurálása
  ADC2_Init();                            // ADC1 konfigurálása
  DMA1_init();
  __enable_irq();                         // Megszakítások globális engedélyezése
  DMA1_ch1_setup();
  ADC1->CR2 |= ADC_CR2_SWSTART;           // Konverzió indítása
  while(!(DMA1->ISR & DMA_ISR_TCIF1));    // Konverzió végére várunk
  DMA1->IFCR |= DMA_IFCR_CGIF1;           // Törli az 1. csatorna jelzöbitjeit
  for(int i=0; i<BLOCK_SIZE; i++) {
    printf("%d\r\n",((DATA_BLOCK[i] & ADC_DR_ADC2DATA) >> 16));
    printf("%d\r\n",(DATA_BLOCK[i] & ADC_DR_DATA));
  }
  while (1) {                             // A program csak egyszer fut le utána
  }                                       // végtelen ciklusban tétlenkedik
}
```

- Az **uart1.c** forrásállományban 4096 bájtos Tx bufferméretet és 115 200 bps adatsebességet állítsunk be!

Program07_6b/main.c – a változtatások

```
void GPIO_Init(void) {
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;       // GPIOA engedélyezés
  GPIOA->CRL &= 0xFFFFFFF0;                 // PA0 analóg bemenet
  RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;       // TIM3 órajel engedélyezés
  RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;       // GPIOB órajel engedélyezés
  /* PB0 konfigurálása T3C3 digitális kimenetként */
  GPIOB->CRL &= ~(GPIO_CRL_MODE0 |
                  GPIO_CRL_CNF0);           // PB0 CNF és Mode bitek törlése
  GPIOB->CRL |=  GPIO_CRL_MODE0_0 |         // PB0 pushpull alt. kimenet, max. 50 MHz
                 GPIO_CRL_MODE0_1 |         // CNF: 10  MODE: 11
                 GPIO_CRL_CNF0_1;
  /* Timer3 CH3 Output Compare mód konfigurálása */
  TIM3->PSC = 0;                            // 0: nincs elöosztás
  TIM3->ARR = 360-1;                        // 5 us-ig számlálunk (half period)
  TIM3->CNT = 0;                            // Számláló nullázása
  TIM3->CCMR2 = TIM_CCMR2_OC3M_0 |          // OC3M=011 output toggle mode
                TIM_CCMR2_OC3M_1;
  TIM3->CCR3 = 0;                           // Egyezési érték
  TIM3->CCER |= TIM_CCER_CC3E;              // CH3 kimenet engedélyezése
  TIM3->CNT = 0;                            // Számláló törlése
  TIM3->CR1 |= TIM_CR1_CEN;                 // Számlálás indítása
}
```


Áldott Karácsonyt


LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot1
Boot0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14
Rasmus Friis Kjeldsen
reblag.dk/stm32
V1.0