AVR-ASM és AVR-C megszakítás (pl.: IT0_OVF) DEBUG-olási lehetősége az AVR STUDIO 4 (Version 4.17) fejlesztőkörnyezet alatt

A.: AVR-ASM kód esetén a megszakítás debug-olásának lépései:

1. AVR STUDIO 4 alatt írjuk meg a szintaktikailag helyes, fordítható, feltehetően működő ASM kódot, melyben megszakítást is használunk (most pl.: 'IT0_OVF'-t)
2. fordítsuk le, és Assemble+Run (CTRL+F7) és (F11) gombok segítségével kezdjük el debug-olni a kódunkat (sárga nyíl megjelenik és lépked)
3. az I/O VIEW ablak automatikusan (vagy kézi beállítás után) megjelenik, melyben keressük meg a használt megszakítás leírását, esetünkben 'TIMER_COUNTER_0'
4. a 'TIMER_COUNTER_0' előtt lévő '+' jelre kattintva, kinyílik egy al-menü, melyben keressük meg a 'Timer/Counter0 Overflow Flag' kifejezést és a mellette levő checkbox-ot
5. ha a futás közben (van sárga nyíl) bármikor a 'Timer/Counter0 Overflow Flag' checkbox-ba egy pipát rakunk, akkor a 'sei' utasítás kiadása után 'megjön' az IT0_OVF a '0x20'-as címen. Melyet látunk is, és egyben elkezdi az IT0_OVF rutint is futtatni és debug-olni (sárga nyíl belemegy az IT-be is, és végrehajtja az ott levő utasításokat)

B.: AVR-C kód esetén a megszakítás debug-olásának lépései:

1. AVR STUDIO 4 alatt írjuk meg a szintaktikailag helyes, fordítható, feltehetően működő C kódot, melyben megszakítást is használunk (most pl.: 'ISR(TIMER0_OVF_vect)'-t
2. ha a változók értékeit nem látjuk, akkor az előzőleg bemutatott dokumentumban szereplő módon, a változókat 'volatile'-ként deklaráljuk. (Lényegében; a C fejlesztői környezet "elrejti" előlünk a hardware-t.)
3. fordítsuk le, és Assemble+Run (CTRL+F7) és (F11) gombok segítségével kezdjük el debug-olni a kódunkat (sárga nyíl megjelenik és lépked)
4. első lehetőségként; lehet, hogy a debug-olás (sárga nyíl) belemegy a megszakítás rutinba is és a változó értékei az ADAT_MEMÓRIÁBAN, a memória végén láthatóvá válnak. (Ha az összes változót 'volatile'-ként deklaráltuk.)
5. második lehetőségként; ha nem megy bele a debug-olás a megszakítás rutinba, akkor az I/O VIEW ablak automatikusan (vagy kézi beállítás után) megjelenik, melyben keressük meg a használt megszakítás leírását, esetünkben 'TIMER_COUNTER_0'
6. a 'TIMER_COUNTER_0' előtt lévő '+' jelre kattintva, kinyílik egy almenü, melyben keressük meg a 'Timer/Counter0 Overflow Flag' kifejezést és a mellette levő checkbox-ot
7. ha a futás közben (van sárga nyíl) bármikor a 'Timer/Counter0 Overflow Flag' checkbox-ba egy pipát rakunk, akkor a 'sei()' utasítás kiadása után 'megjön' az IT0_OVF a '0x20'-as címen. Melyet látunk is, és egyben elkezdi az 'ISR(TIMER0_OVF_vect)' megszakítást futtatni és debug-olni (sárga nyíl belemegy az IT-be is, és végrehajtja az ott levő utasításokat)

AVR-ASM (egyszerű) mintaprogram az IT debug-olás kipróbálásához:

```
;===============================================================================
; 6. FELADAT: Futófény megvalósítása IT0 MEGSZAKÍTÁS-sal
;-------------------------------------------------------------------------------
;              Megszakítás, Interrupt, IT0 működés bemutatása
;
;              LED futófény megvalósítása IT0-ás INTERRUPT-tal
;===============================================================================

.include "m128def.inc"
.def tmp =r16          ; segédváltozó
.def led =r18          ; LED meghajtó regiszter
.def tmp_s=r19         ; STACK mentésére szolgáló segédváltozó (opcionálisan használható)
;---------------------------------------------------------------

.org 0x0               ; Ezen a címen indul a PROGI
       rjmp start

.org 0x20
       rjmp IT0_OVF    ; IT0 Overflow megszakítás bejövetelének jelzése az 0x20-as címen van,
                       ; ha a IT0 8 bites számlálója túlcsordul, akkor ad egy IT-t 0x20 címre.
.org 0x100
start:
                       ; --- STACK inicializálása ------------------
       ldi     tmp, HIGH(RAMEND)      ; stack beállítás
       out     SPH, tmp               ; a memória végére teszi
       ldi     tmp, LOW(RAMEND)       ; a stack pointer-t
       out     SPL, tmp
       clr     tmp

                       ; --- LED-eket vezérlő PORT-ok beállítása PORTB, PORTD ---
       ldi tmp,    0b11110000 ; PORT-ok felső 4 bitje kimenetre állítani
       out DDRB,   tmp
       out DDRD,   tmp

       ldi led, 0b0000_0001  ; LED változóba a kezdeti szám
       clc
                       ; --- IT0 kezdeti beállítása ----------------
       ldi tmp,        0b0000_0111    ; segédváltozó a beállításhoz
       out TCCR0,      tmp            ; 105. oldal DSh.-ből, CLK/1024 előosztást állítok be
       ldi tmp,        0b0000_0001
       out TIMSK,      tmp            ; IT0-ás INTERRUPT engedélyezést állítok be, ha az IT0
                                      ; 8 bites számlálója túlcsordul, akkor ad INTERRUPT-ot

       sei                            ; proci engedi befogadni a bejövő IT-t
                       ; --- IT0 kezdeti beállítás VÉGE ----------------

CIKLUS:                ; ITT PÖRÖG A PROGI, látszólag nem csinál semmit
              sleep    ; kisfogyasztású állapot
       jmp CIKLUS


; --- Megszakítás IT0 ------------------------
                       ; LÁTHATÓ, hogy az IT0-át nem a FŐprogramból hívjuk meg, hanem egy külső
                       ; HW kezdeményezi, és utána, ha a PROCI elfogadja a megszakítást (SEI),
                       ; akkor lefut az IT0-hoz tartozó IT rutin.

IT0_OVF:
                       ; CSAK minden 16. IT megjövetelekor lép a LED       (LASSÍTÁS)
       ; dec tmp      ; 1-el csökkentjük a tmp értékét     (LASSÍTÁSHOZ KIVENNI a kommentet)
       ; brne cimke1  ; ugrik a cimke1-re, ha tmp nem "0"  (LASSÍTÁSHOZ KIVENNI a kommentet)

       ; ldi tmp, 16  ; ha tmp=0, akkor belemegy és egyből kezdeti értéket kell adni neki,
                       ; itt tmp=16 (LASSÍTÁSHOZ KIVENNI a kommentet)

       out  PORTD, led       ; kitenni a LED változó felső 4 bitjét, a PORTD felső 4 bitjére
       swap led              ; felső 4 és alsó 4 bit csere
       out  PORTB, led       ; kitenni a LED változó eredeti alsó 4 bitjét, a PORTB felső
                             ; 4 bitjére (LED meghajtás)
       swap led              ; visszaállítjuk a számot, hogy utána tudjuk léptetni

       rol     led           ; LED, ROTATE LEFT th. CARRY
cimke1:

reti
```

AVR-C (egyszerű) mintaprogram az IT debug-olás kipróbálásához:

```
/*
==== 2222 2222 2222 2222 ================================================

       2. feladat:

          IT0 - MEGSZAKÍTÁS kezelése
          LED-ek villogása IT0 megszakítással megoldva (IT0-val)

          FIGYELEM; IT0 nem osztható pontos másodperces időegységre:
                      8 MHz/1024/256 = 30,5175 [ütés/sec]
                     16 MHz/1024/256 = 61,0351 [ütés/sec]

-----------------------------------------------------------------
*/

#include <avr/io.h>
#include <avr/interrupt.h>                  // INTERRUPT kezeléshez ez kell

volatile unsigned char szamlalo = 0;        // IT0 rutin-ban használt globális változó


int main()
{
       DDRB  = 0xF0;                        // uC beállítások
       DDRD  = 0xF0;                        // PORTB és PORTD felső 4 bitje kimenet, LED-ek

       PORTD = 0x00;                        // PORTD kezdeti értéke (semelyik LED nem világít)
       PORTB = 0x00;                        // PORTB kezdeti értéke (semelyik LED nem világít)

       TCCR0 = 0b00000111;                  // T0 interrupt 1024-es előosztás beállítása
       TIMSK = 0b00000001;                  // T0 interrupt OVF engedélyezése

       sei();                               // Processzor elfogadja az interrup-ot

       while(1){};                   // kezdeti beállítás után itt pörög a program! ÜRES.

return (0);
};


// --- IT0 interrupt kezelés --- --- --- --- --- --- ---

ISR(TIMER0_OVF_vect)                        // IT0_OVF interrupt megjövetele
{
       if (szamlalo == 15)                  // villogás „lassítás” (8MHz:"15", 16MHz:"30",
                                            // akkor kb. 1 sec-es a LED),
       {                                    // ha kell MINDEN "xx"-edik INTERRUPT MEGY BELE
                                            // az IF-es ÁGBA (FONTOS)

              PORTD=PORTD^0b11110000;       // PORTD LED lábait XOR-ozzuk (T-ff)
              PORTB=PORTB^0b11110000;       // PORTB LED lábait XOR-ozzuk (T-ff)

              szamlalo = 0;                 // számláló visszaállítása
       };

       szamlalo++;
}
```