ÓBUDAI EGYETEM

TIMER modul működésének áttekintése

Modul VI.

Lecke célja

- A lecke célja bemutatni a beágyazott rendszereknél használt időzítési lehetőségeket.
- Bemutatásra kerül az Arduino IDE által biztosított delay(), delayMicroseconds(), millis(), micros() függvények, továbbá egy timer modul regiszter szintű konfigurálása is.

Bevezetés

- A beágyazott rendszerek egyik legfontosabb perifériája az úgynevezett TIMER (időzítő).
- Működés során sokszor valamilyen ciklikus feladatot kell elvégeznie a rendszernek. Ez lehet fix időintervallumonkénti adatgyűjtés, megjelenítés, beavatkozás, vagy két interakció között eltel idő mérése stb.
- A TIMER egyik fő feladata, hogy ezeket az időzített, ütemezett feladatokat menedzselje.
- Külön perifériaként állnak rendelkezésre a mikroprocesszoron belül.

Bevezetés

- A TIMER alapja egy számláló regiszter. A számláló regiszter a bejövő impulzusok változásának a függvényében inkrementálja vagy dekrementálja az értékét.
- Ha ismert az impulzusváltozás frekvenciája, pl. egy fix órajel generátorral (oszcillátorral) egészítjük ki a számláló regisztert, akkor beszélhetünk időzítőről (TIMER).
- A órajel frekvenciájának és a számláló értékének az ismeretében könnyedén fel lehet használni ütemezésre ezt az összeállítást.
- Pl. egy 1MHz-es órajel esetén, a periódusidő 1/1000000 másodperc = 1us időközönként inkrementálódik a számláló érétke

Bevezetés

- Az ATmega2560 mikroprocesszor több TIMER-el is rendelkezik.
- Két csoport érhető el az adatlap alapján:
  - 8bites TIMER 0/2
  - 16 bites TIMER 1/3/4/5
- A bitek száma itt a számláló regiszterre vonatkozik.
- Természetesen nem csak időzítési feladatra használható fel egy TIMER modul, hanem különböző jelek (pl. PWM) generálására is, frekvenciagenerátorként lehet alkalmazni, külső események számlálása stb. (típus függő). Azonban ez most nem a lecke célja.

Millis(), micros()

- Az Arduino IDE által biztosított millis() és micros() függvényeket időmérésre lehet felhasználni.
- A függvény nevében is látható a különbség. Az egyikkel milliszekundumban a másikkal pedig mikro szekundumban eltelt időt lehet mérni.
- A függvények a TIMER0 időzítőre építenek.
- A visszatérési érték unsigned long típusú.
- Maximális értékből meghatározható, hogy:
  - A millis() kb. 49 nap után túlcsordul
  - A micros() kb. 70 perc után túlcsordul

delay(), delayMicroseconds()

- A delay() és delayMicroseconds() függvényeket késleltetés esetén lehet használni.
- Paraméterként milli vagy mikroszekundumban kell megadni a késletetési időt.
- A legnagyobb hátránya mind a két függvénynek, hogy blokkolnak! Ami azt jelenti, hogy amíg a megadott idő el nem telik, a processzor áll, nem „végez hasznos munkát”.
- Használata egyszerűbb programok esetén nem probléma, azonban ha lehet kerülni kell!

Időmérés

```
1  unsigned long t_start=0;
2  unsigned long t_stop=0;
3  unsigned long num=0;
4
5  unsigned long factorial(int n);
6
7
8  void setup()
9  {
10   Serial.begin(9600);
11
12 }
13
```

Időmérés

```
27 unsigned long factorial(int n){
28   if (n==0){
29     return 1;
30   }else{
31     unsigned long fact=1;
32     for (int i=1; i<n; i++){
33       fact=fact*i;
34     }
35     return fact;
36   }
37 }
```

Időmérés

```
15 void loop()
16 {
17   t_start=micros();
18   num=factorial(10);
19   t_stop=micros();
20   Serial.print(num);
21   Serial.print(", time: ");
22   Serial.print(t_stop-t_start);
23   Serial.println("us");
24   delay(5000);
25 }
```

Nem blokkoló időzítés

```
1  #define LED 13
2  unsigned long t =0;
3  unsigned long prev_t_LED=0;
4  unsigned long prev_t_UART=0;
5
6  void setup()
7  {
8      pinMode(LED, OUTPUT);
9      Serial.begin(9600);
10     t = millis();
11     prev_t_LED=t;
12     prev_t_LED=t;
13 }
14
```

Nem blokkoló időzítés

```
16 void loop()
17 {
18   t=millis();
19   if (t-prev_t_LED>1000){
20     digitalWrite(LED, 1-digitalRead(LED));
21     prev_t_LED=prev_t_LED+1000;
22   }
23
24   if (t-prev_t_UART>3000){
25     Serial.println("Hello world!");
26     prev_t_UART=prev_t_UART+3000;
27   }
28 }
```

Timer

- Az előző két példában bemutatásra került, hogyan lehet eltelt időt mérni, illetve a blokkoló késleltetéseket elkerülni.
- A bemutatott példában előfordulhat olyan szituáció, amikor a loop ciklus újrakezdése több időig tart (pl. bonyolult számítás végzése miatt) mint az általunk elvárt legrövidebb ütemezési feladat periódusideje.
- Ebben az esetben az időzítést egy megszakításba kell kiszervezni.
- Minden TIMER túlcsordulás esetén egy megszakítással jelzi ezt az eseményt (programozható). A megszakítást lehet felhasználni ütemezésre.
- Azonban itt is figyelembe kell venni, hogy hosszú, számításigényes feladatot ha lehet, megszakításon belül ne végezzünk, mert addig blokkolva van a processzor!

TIMER 5

- A következőekben a TIMER 5 beállítása kerül bemutatásra.
- Időzítési feladathoz a következő regiszterekre lesz szükség:
  - TCCR5A
  - TCCR5B
  - TCNT5
  - TIMSK5
  - TOIE5
- A TIMER modult a regiszterek egyes bitjeinek a bebillentésével lehet beállítani. Minden egyes bithez tartozik valamilyen beállítás. (adatlap!!!)

TCCR5A és TCCR5B regiszter

- A TCCRnA/B (Timer/Counter Control Regisister) regiszterrel lehet a modult beállítani.
- A mi esetünkben egy sima időzítési feladatot fog ellátni.
- A TCCR5B regiszterben található CS:52-CS:50 bitekkel lehet bekapcsolni és kikapcsolni (órajelet beállítás) a modult.

17.11.8 TCCR5B – Timer/Counter 5 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x121)	ICNC5	ICES5	–	WGM53	WGM52	CS52	CS51	CS50	TCCR5B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR5A és TCCR5B regiszter

- A TIMER5 beállítástól függően a CPU fő órajelét, osztással/osztás nélkül (16MHz) vagy külső órajel forrást használhat. Ezt „CS” bitekkel lehet beállítani az adatlapban megtalálható értékek függvényében:

Table 17-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

TIMER5

- A fő órajelet leosztva lehet szabályozni a számláló növekedésének periódusidejét.
- A maximális értéket a TCNT5 regiszter szabja meg. A TCNT5 egy 16 bites regiszter, azaz 65535 lehet a maximális érétke.
- A közvetlenül írható és olvasható.
- Külön kezelhető az alsó és felső 8 bit.(TCNT5H és TCNT5L)

17.11.16 TCNT5H and TCNT5L –Timer/Counter 5

Bit	7	6	5	4	3	2	1	0	
(0x125)	TCNT5[15:8]								TCNT5H
(0x124)	TCNT5[7:0]								TCNT5L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TIFR5 regiszeter

- Túlcsordulás esetén a TIFR5 (Timer/Counter5 Interrupt Flag Regiszter) regiszterben található TOV5 (Timer/Countern, Overflow Flag) jelzőbit állapota billen át 1 érékre.
- Ezt az állapot változást lehet felhasználni ütemezésre. A jelzőbithez hozzá lehet rendelni egy megszakítást, ami akkor keletkezik, ha 1 lesz az értéke.
- Ezt a TIMSK5 regiszter segítségével lehet aktiválni.

17.11.40 TIFR5 – Timer/Counter5 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	–	–	ICF5	–	OCF5C	OCF5B	OCF5A	TOV5	TIFR5
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TIMSK5 regiszter

- A TIMSK5 regiszter legkisebb helyiértékű bitjéhez rendelt TOIE5 (Timer/Countern, Overflow Interrupt Enable) a bebillentésével lehet engedélyezni vagy tiltani a túlcsorduláshoz tartozó megszakításkérést.
- A kiváltott megszakításneve TIMER5_OVF_vect.

17.11.36 TIMSK5 – Timer/Counter 5 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x73)	–	–	ICIE5	–	OCIE5C	OCIE5B	OCIE5A	TOIE5	TIMSK5
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Folyamat rövid áttekintése

ÓBUDAI EGYETEM

7.4.1 SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the “Instruction Set Summary” on page 404.

Folyamat rövid áttekintése

- 65535 után a TCNT5 regiszter túlcsordul és a TIFR regiszterben található TOV5 bit bebillen. Mivel a TIMSK5 regiszterben TOIE5 engedélyezve van (és természetesen az SREG regiszter általános megszakításokhoz tartozó „I” (Global Interrupt Enable) is engedélyezve van) akkor a CPU végrehajtja az ISR (Interrupt Service Routin) rutint.
- A kiszolgálás után a TOV5 automatikusan törlődik és a TCNT5 regiszter elkezd 0-tól újra számolni.

1 másodperces ütemező

- Az időzítést a TCNT5 regiszter értékének a manipulálásával lehet beállítani.
- Ismert a TIMER5 modulra kapcsolt órajel, azaz az inkrementálás periódus ideje.
- A feladat egy 1s-os ütemező létrehozása, 1024-es órajel leosztás esetén:
  - TIMER órajele: 16Mhz/1024= 15625Hz
  - Periódusidő: 1/15625 = 0.000064 s
  - 1s időzítő -> 1s/ 0.000064s =15625
  - A TCNT5 számláló érétkét tehát 65535-15625= 49910-ről kell indítani, ekkor 1s-ként fog túlcsordulni, azaz megszakítást eredményezni.

1 másodperces ütemező

```
1  #define LED 13
2
3  void InitTimer5();
4
5  void setup()
6  {
7    pinMode(LED, OUTPUT);
8    InitTimer5();
9    }
10
11 void loop()
12 {}
```

1 másodperces ütemező

```
14 void InitTimer5(){
15   TCCR5A=0;
16   TCCR5B=0;
17   TCCR5B=B00000101; //1024 prescaler
18   TCNT5=65535-15625;
19   TIMSK5 |= (1<<TOIE5);
20 }
```

1 másodperces ütemező

```
22 ISR(TIMER5_OVF_vect){
23   digitalWrite(LED, 1-digitalRead(LED));
24   TCNT5=65535-15625;
25 }
```