

Labor gyakorlat – Mikrovezérlők

ATMEL AVR – ARDUINO

1. ELŐADÁS

BUDAI TAMÁS

2016. 09. 14.

Labor 2 – mikrovezérlők modul

- 2 alkalom
 - 1 mikrovezérlők felépítése, elmélet
 - 2 programozás, mintaprogramok
- **Értékelés:** a 2. alkalom elején miniZH az 1. alkalommal elhangzottakból.
Forma: 5+1 rövid választ igénylő kérdés

- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

Mikrovezérlő:

- Kisméretű, alacsony fogyasztású, programvezérelt digitális hálózat, avagy **számítógép**



Alkalmazási területek:

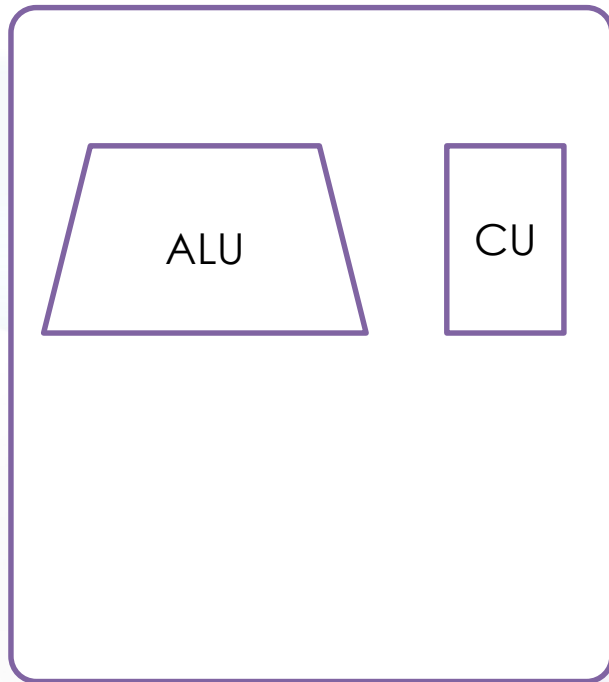
- Háztartási és szórakoztató elektronika
- Ipari elektronika
- Járműipar
- Gyógyászati eszközök
- „SMART” eszközök
- IoT
- ...



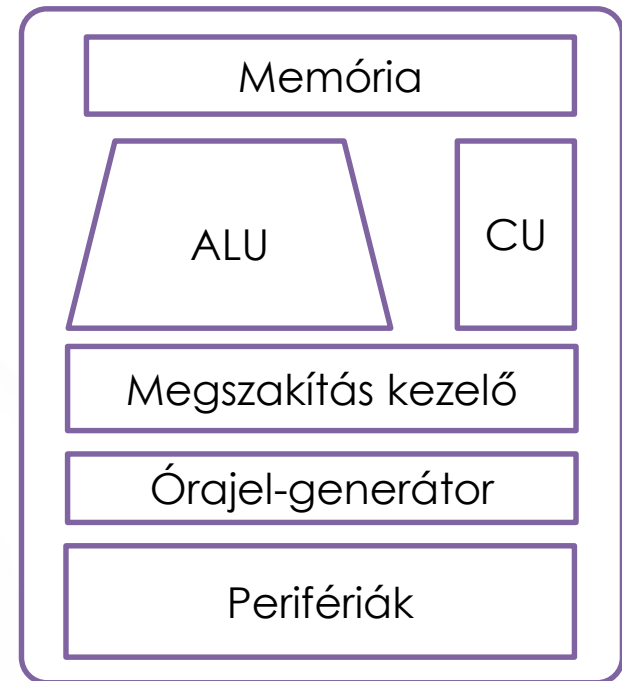
- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

Mikrovezérlők felépítése, működése

Mikroprocesszor



Mikrovezérlő

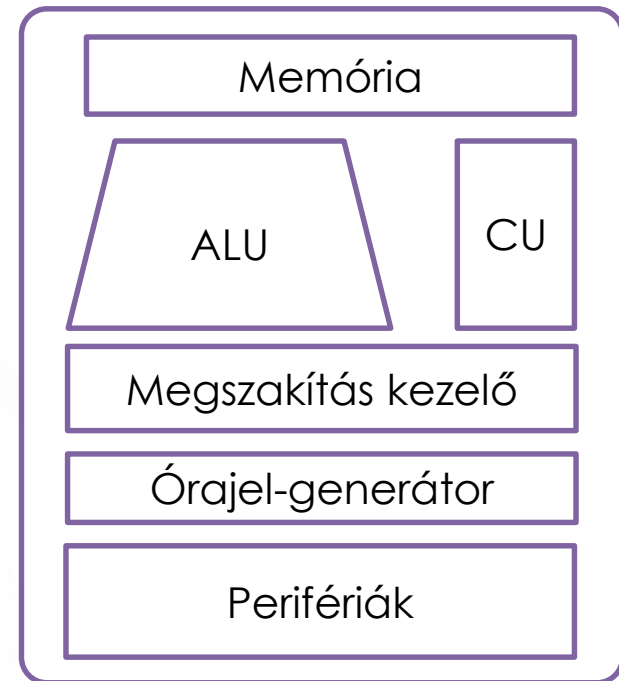


A mikrovezérlő egy teljes számítógép egy tokban

Számítógép, tehát
érvényesek rá a Neumann
alapelvek:

- Elektronikus
- Bináris számrendszer
- Adat és programmemória
- Univerzális Turing-gép

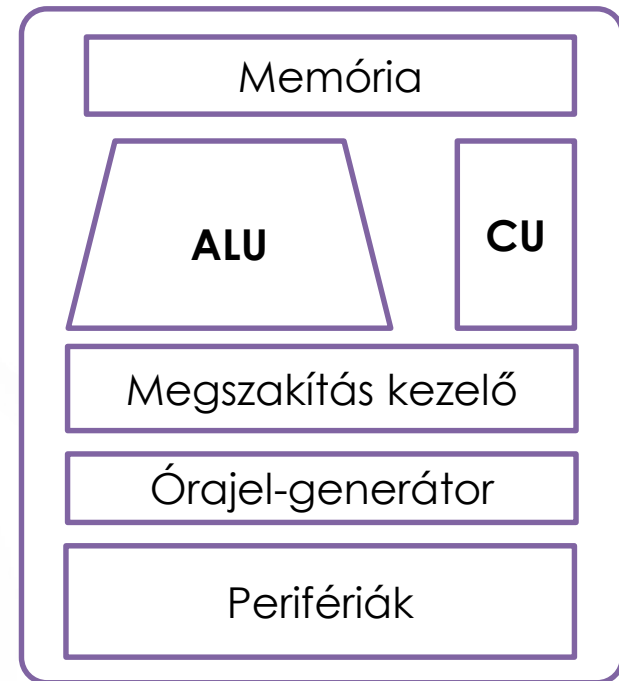
Mikrovezérlő



CPU (ALU+CU):

- Egyes családoknál* közös
- Meghatározza a számaábrázolást*:
8,16,32bit...

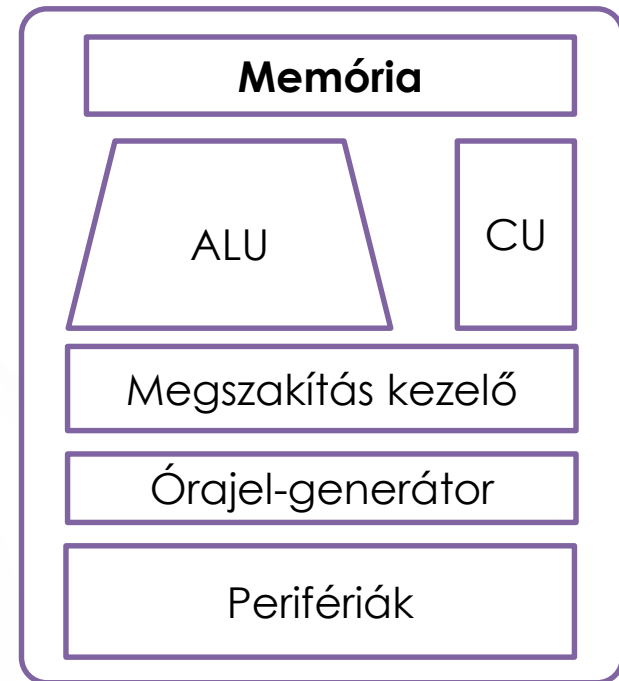
Mikrovezérlő



Memória típusok:

- Program memória
 - nem újraírható
 - újraírható
- Adatmemória
 - RAM
 - ROM

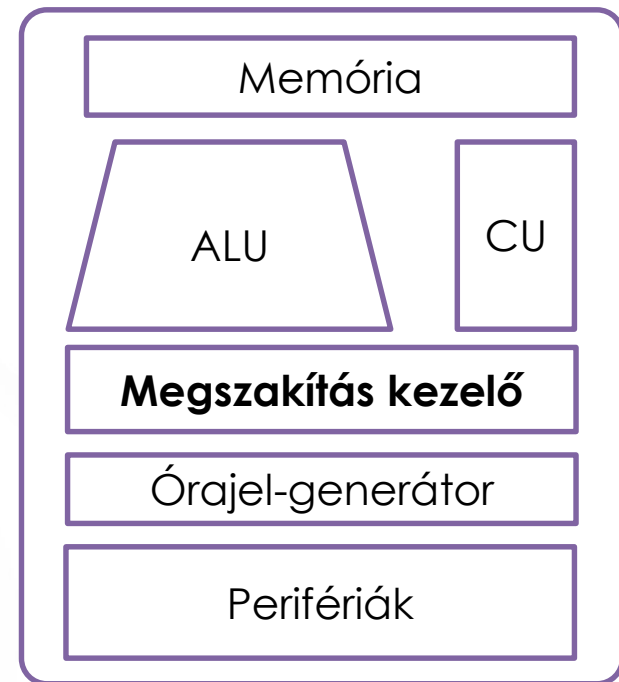
Mikrovezérlő



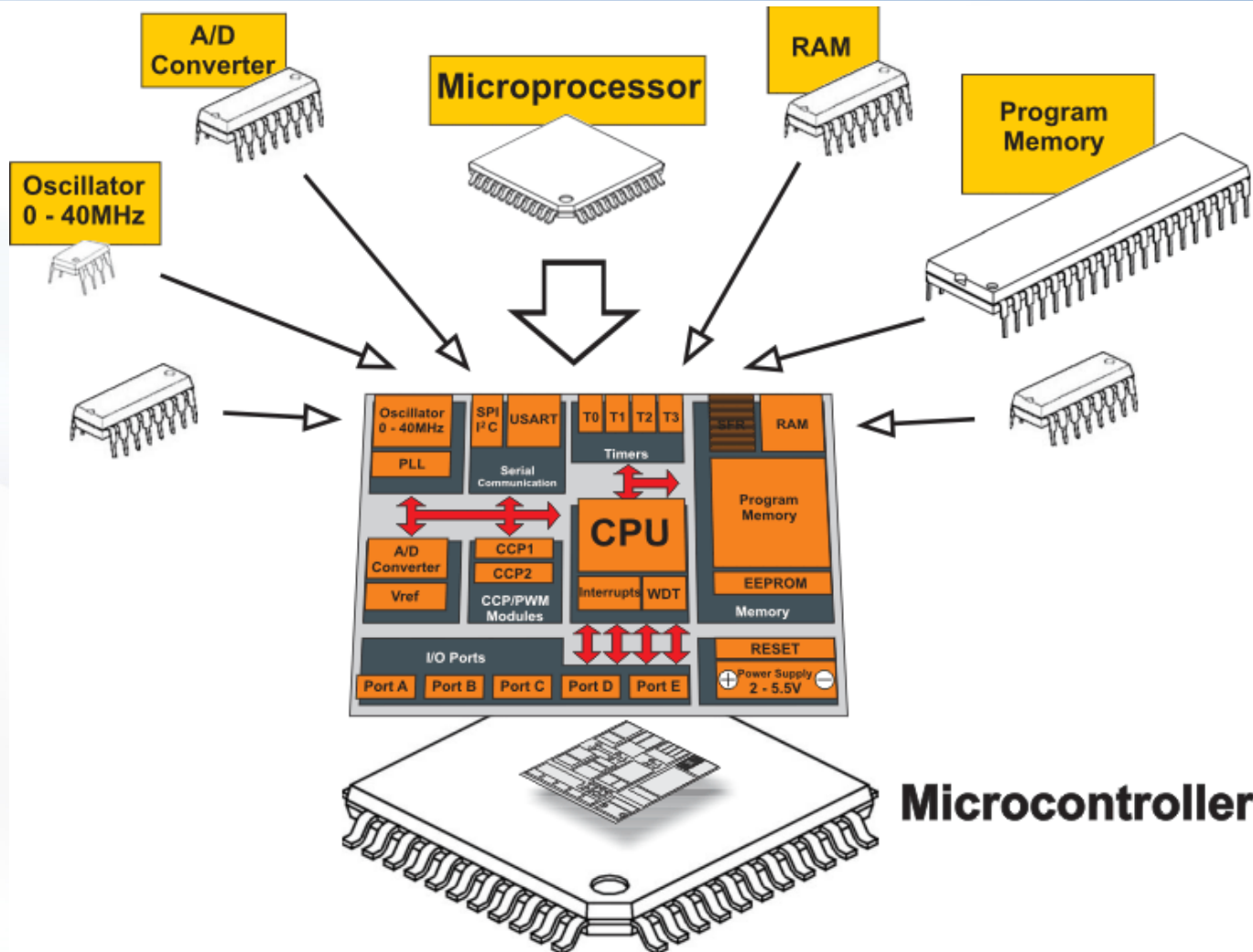
Megszakítás kezelő:

- A normál futást megszakítva rákényszeríti a rendszert, hogy azonnal hajtson végre egy speciális programrészt
- Megszakítás forrásai
 - **Belső:** időzítő, periféria
 - **Külső:** adott lábon felfutó él

Mikrovezérlő



Mikrovezérlők felépítése, működése



Működés

Egy általános célú mikrovezérlő a **reset** esemény után a következő lépéseket hajtja végre:

1. Minden periféria alaphelyzetbe állítása
2. A **program** lépéseinek végrehajtása a programmemória elejétől (reset vector) az **órajel** szerint

Reset

A **reset** esemény: a mikrovezérlő (újra)indítása

A reset esemény forrásai:

- Power-on reset (bekapcsolás)
- External reset (külső reset láb szintváltozása miatt)
- Watchdog system reset (őrkutya* miatt)

*Ezért a fordításért külön elnézést kérek

Program (Gépi kód)

- A gép által értelmezhető binárisan kódolt **utasítások** sorozata

Forráskód

- Majd a programozásnál...

Órajel

- Mivel a mikrovezérlő egy digitális hálózat ezért a működéséhez szükség van **órajelre**.
- Jelalak: **négyszögjel**
- mértékegysége: 1/s vagyis **Hertz**
- Tipikus nagyságrendje napjainkban: 1MHz-100MHz

Forrás	Pontosság	Költség
RC oszcillátor	Alacsony	Legolcsóbb
Kerámia rezonátor	Közepes	Olcsó
Rezgőkristály	Magas	Drága

Órajel ciklus, ciklusidő

- **Ciklusidő:** Az órajel két egymást követő le/felfutó éle közt eltelt idő (vagyis a négyszögjel **periódusideje**)
- Utasítások jellemzése: hány órajelciklus szükséges a végrehajtáshoz

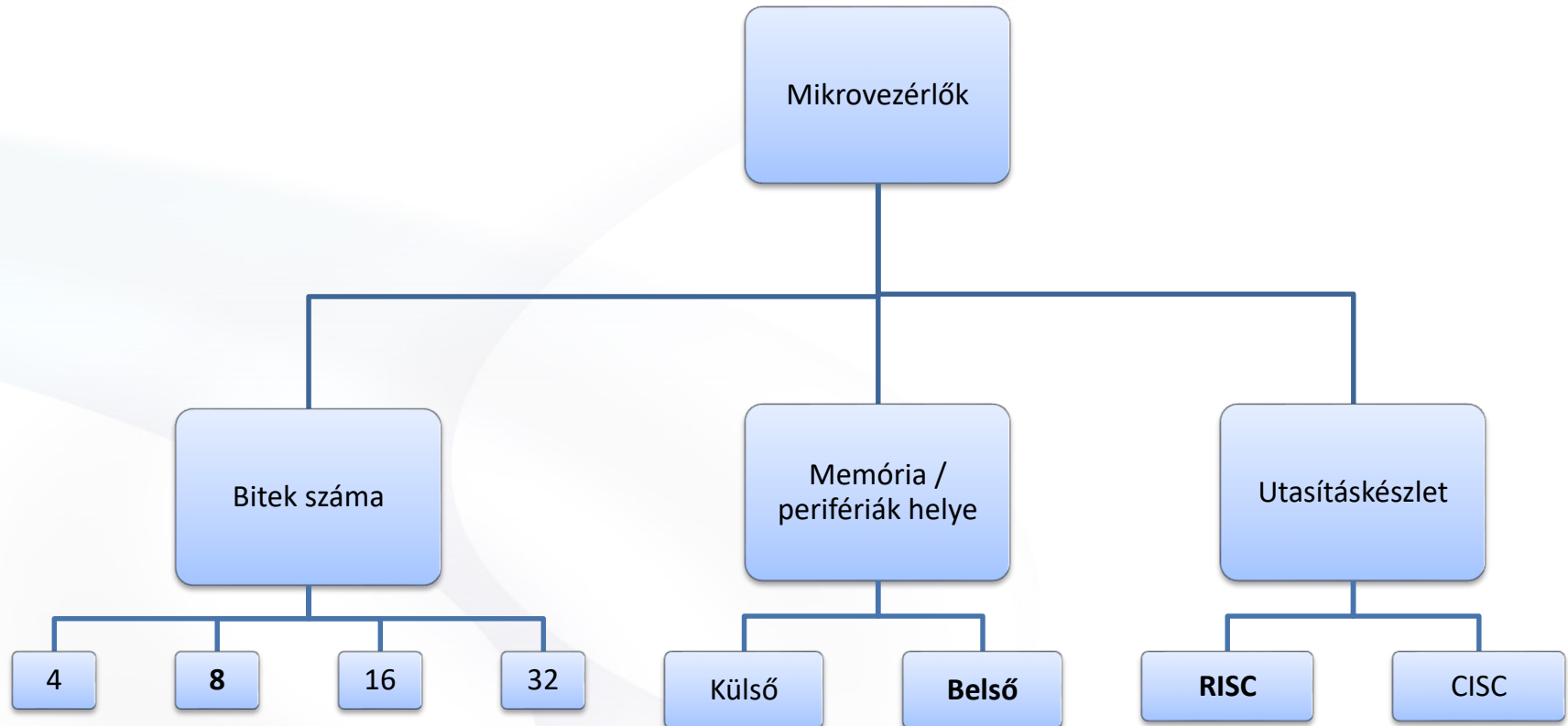
Pl.:

- Feltétel nélküli ugrás – 1 ciklus
- Összeadás – 2 ciklus

(Minél kevesebb, annál gyorsabb)

- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

Mikrovezérlő típusok, gyártók



Gyártók

- Atmel
- Infineon
- Intel
- Microchip
- Motorola
- National Instruments
- Parallax
- Texas instruments
- Xilinx
- ...



MICROCHIP



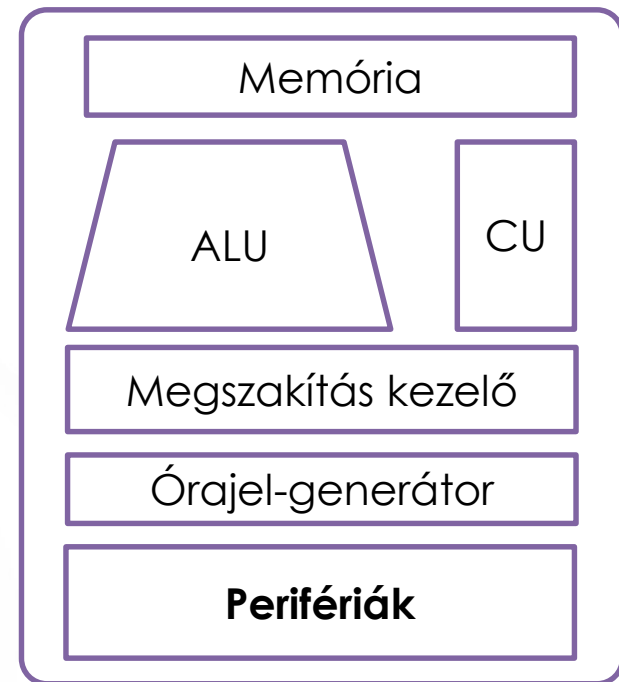
PARALLAX 
www.parallax.com



- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

- **Tipikus perifériák**
 - Digitális be/kimenetek
 - A/D konverter
 - D/A konverter
- **Speciális perifériák**
 - BUS illesztők*:
 - SPI
 - UART (RS232)
 - I²C
 - CAN
 - ...
 - érintés-érzékelő
 - PWM kimenet

Mikrovezérlő



*Nem feltétlenül szükséges minden buszhoz (bitbang)

- **Digitális bemenet:**
 - Egy bemenet egy láb
 - Egy bit reprezentálja a logikai szintet:
 - 0 = alacsony
 - 1 = magas
- **Digitális kimenet:**
 - Egy kimenet egy láb
 - Ugyan az, mint a bemenet, csak itt a bitet írjuk és nem olvassuk

- **A/D átalakító:**
 - Bemenet
 - Működés:
 - Mintavételezés
 - Kvantálás
 - Jellemzői:
 - Felbontás (hány bites)
 - sebesség
 - Az analóg jel amplitúdója a föld és egy referenciapotenciál közt értelmezett (ez általában a tápfeszültség) és egy változóból kiolvasható

- **D/A átalakító:**

- Kimenet
- Jellemzői:
 - Felbontás (hány bites)
 - sebesség
- Az analóg jel amplitúdója a föld és egy referenciapotenciál közt értelmezett (ez általában a tápfeszültség)
- Logikája azonos az A/D-vel, csak itt beírjuk a kívánt szintet és nem kiolvassuk

- **BUSZ illesztők:**
 - Be/kimenet
 - Annyi lábat használ, amennyit az adott BUSZ megkíván:
 - Pl.: CLK, RX, TX
 - Működését az adott BUSZ típusa határozza meg
 - Tartozhat hozzá speciális memóriarész (pl. fogadó buffer)

- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

Program (Gépi kód)

- A gép által értelmezhető binárisan kódolt **utasítások** sorozata

Forráskód

- Az egyszerűbb kezelhetőség érdekében adjunk rövid nevet az egyes utasításoknak
- Ember által értelmezhető, gép által **nem**;
- Ahhoz, hogy futtatható legyen **fordításra** van szükség!

Forráskód

```
#include <stdio.h>

#define ONE 1

int main()
{
    int j=5;
    int tanult = 1;
```

Fordító

Egy mikrovezérlő bármilyen nyelven programozható, ha van az adott nyelvhez és vezérlőhöz fordítóprogram.

Gépi kód

```
1010010
1110101
0101010
```

- **Programnyelvek**
 - Alacsony szintű
 - Assembly
 - Magas szintű
 - **C**, C++, Python, Java...

- **Mikrovezérlők**
 - Mikrovezérlők felépítése, működése
 - Mikrovezérlő típusok, gyártók
 - Mikrovezérlők perifériái
- **Mikrovezérlők programozása**
 - A C programozási nyelv (ismétlés)
 - ATMEL AVR mikrovezérlők programozása
 - Az ATMEL studio használata
 - Feladatmegoldás

- Alapok
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

A forráskód:

- Egyszerű szövegfájl
- Ember által értelmezhető
- Utasítások sorozata
- **Kötött formátum**

Példa

```
1  #include <stdio.h>
2
3  #define ONE 1
4
5  int main()
6  {
7      int j=5;
8      int tanult = 1;
9
10     if(tanult){
11         printf("Az on erdemjegye: %d\n", j);
12     }
13     else{
14         printf("Az on erdemjegye: %d\n", ONE);
15     }
16
17     return 0;
18 }
19
```

#include direktíva:

- más forráskódrészeket tudunk beágyazni
- ezek erősen függenek a használt környezettől

main() függvény:

- minden programban pontosan egy darabnak kell lennie
- itt kezdődik a programunk végrehajtása

A C programozási nyelv (ismétlés)

Változók, adattípusok

A legkisebb egység: 1 bit

B

8 bit = **1 byte**

B

B

B

B

B

B

B

B

Változótípusok

void	„semmi” (ezt is jelölni kell valahogy)
bool	logikai (igaz/hamis)
char	egy karakter (1 byte)
int	(előjeles) egész szám
unsigned int	egész szám
float	lebegőpontos szám
double	dupla pontosságú float
string	char-okból álló tömb

A C programozási nyelv (ismétlés)

Változók, adattípusok

Példa

```
1  int a;  
2  float b;  
3  double c;  
4  
5  a=5;  
6  b=3.14;  
7  c = 6.53;  
8  
9  int n = 0;  
10 float x = 1.2345;
```

Tömbök:

- Egy adott adattípusból álló több elemű halmaz
- A memóriában folytonosan helyezkedik el
- A tömb méretét előre meg kell mondanunk*

Tömbök:

- Egy adott adattípusból álló több elemű halmaz
- A memóriában folytonosan helyezkedik el
- A tömb méretét előre meg kell mondanunk*

```
1 int ints[6];  
2 int pins[] = {2, 4, 8, 3, 6};  
3 int vals[6] = {2, 4, -8, 3, 2};  
4  
5 printf("%d", pins[0]);  
6 printf("%d", vals[3]);
```

Operátorok:

- Aritmetikai: + - * / % ()
- Összehasonlító: == != < <= > >=
- Logikai: && || !
- Bitenkénti: & | ^ ~ << >>
- Helyben módosító: ++ -- += -= *= /= &= |=
- Egyéb: sizeof

A C programozási nyelv (ismétlés)

Operátorok

Példa

```
1  int a,b,c;  
2  int d;  
3  int e;  
4  
5  a=5;  
6  b=2;  
7  c=3;  
8  
9  d = a+b*c;  
10  
11 d= (a-b)*c;  
12
```

Elágazások:

- if, if-else
- switch case

A C programozási nyelv (ismétlés)

Elágazások, feltételes utasítások

If-else

```
1  int x=0;
2  int val = 700;
3
4  if (val < 500)
5  {
6      x = 1;
7  }
8  else
9  {
10     x = 0;
11 }
```

Switch-case

```
1 int keres;  
2 keres = 2;  
3  
4 switch (keres) {  
5     case 1:  
6         printf("Elso");  
7         break;  
8     case 2:  
9         printf("Masodik");  
10        break;  
11    default:  
12        printf("Egyik sem");  
13 }
```

Ciklusok:

- for
- while
- do-while
- Ciklusvezérlés

for

```
1 int i;  
2 i = 0;  
3  
4 for (; i <= 255; i++){  
5     printf("%d ", i);  
6 }
```

Tömörebben:

```
1 for (int i=0; i <= 255; i++){  
2     printf("%d ", i);  
3 }
```

A C programozási nyelv (ismétlés)

Elágazások, feltételes utasítások

while

```
1 int var = 0;
2 while(var < 200){
3     printf("%d", var);
4     var++;
5 }
```

```
1 int var = 400;
2 while(var > 200){
3     printf("%d", var);
4     var--;
5 }
```

do-while

```
1 int x = 0;
2 do
3 {
4     printf("Ertek: %d",x);
5     x++;
6
7 } while (x < 100);
```

Vezérlő utasítások:

- break
- continue
- return

- break

```
1  int var = 0;
2  while(var < 200){
3
4      if(var>10){
5          printf("elertuk a limitet!");
6          break;
7      }
8
9      var++;
10 }
```

A C programozási nyelv (ismétlés)

Ciklusok

- break

```
1  int kerdes;  
2  kerdes = 2;  
3  
4  switch (kerdes) {  
5      case 1:  
6          printf("Elso");  
7          break;  
8      case 2:  
9          printf("Masodik");  
10         break;  
11     default:  
12         printf("Egyik sem");  
13 }
```

- continue

```
1 int var = 0;
2 while(var < 200){
3     var++;
4
5     if((var%2) == 0){
6         continue;
7     }
8
9     printf("%d ",var);
10
11 }
```

A C programozási nyelv (ismétlés)

Ciklusok

- return

```
1 ▾ if(err != 0){  
2     printf("HIBA!");  
3     return 1;  
4 }
```

```
1  #include <stdio.h>  
2  
3  #define ONE 1  
4  
5  int main()  
6  {  
7      int j=5;  
8      int tanult = 1;  
9  
10 ▾ if(tanult){  
11     printf("Az on erdemjegye: %d\n", j);  
12 }  
13 ▾ else{  
14     printf("Az on erdemjegye: %d\n", ONE);  
15 }  
16  
17     return 0;  
18 }  
19
```

Függvények

- mint a matematikában:
 - paraméter, visszatérési érték
- programszervezésre használhatók

Példa: $y = \sin(x)$

Példa

```
1  #include <stdio.h>
2  #include <math.h>
3
4  float sinfok(int f){
5      return sin(f*M_PI/180);
6  }
7
8
9  int main(){
10     int x = 30;
11     printf("%f\n", sinfok(x));
12 }
```

Összefoglalás:

= + - * / % () == != < <= > >= ++ -- += -=
*= /= &= |= sizeof bool int float [] {} ;
#include

Mit hagytunk ki:

- előfordító direktívák, makrók
- mutatók, mutató aritmetika
- dinamikus memóriakezelés
- saját adattípusok, struktúrák
- függvénykönyvtárak
- hibakeresés
- ...

Ajánlott irodalom:

- <http://www.tutorialspoint.com/cprogramming/index.htm>
- B. W. Kernighan - D. M. Ritchie : A C programozási nyelv
- Takács Gábor: Programozás villamosmérnököknek