EduDaq 0.0

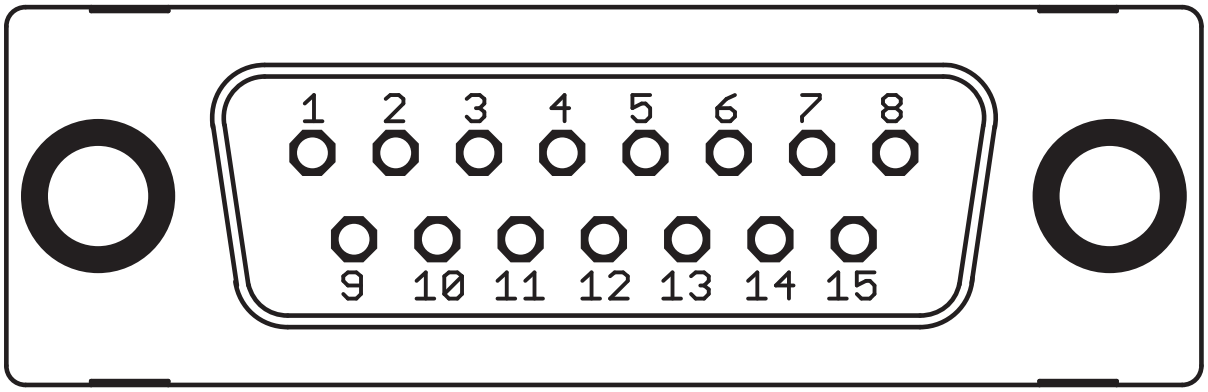
1. Be- és kimeneti csatornák

Az eszköz két 16-bites analóg-digitális átalakítót tartalmaz, ezeket a továbbiakban ADC_1-gyel és ADC_2-vel jelöljük. Az `@A` paranccsal választható ki, hogy melyikük aktív.

Az eszköz 4 bemeneti csatornát képes kezelni, ezekre **A, B, C, D** jelölésekkel hivatkozunk a későbbiekben. Mindkét analóg-digitális átalakító esetében két-két csatorna közül választhatunk: az ADC_1-hez az **A** és **B** csatornák közül az egyik rendelhető hozzá az `@1` paranccsal, míg az ADC_2-hez a **C** és **D** egyike társítható az `@2` paranccsal.

A műszer feszültség kiadására is képes. Két 12-bites digitális-analóg átalakítót tartalmaz, ezek jele DAC_1 és DAC_2 a továbbiakban. Az átalakítók feszültségének beállítása a `@d` és `@D` parancsokkal történik.

2. Csatlakozó



1. ábra. Az EduDaq csatlakozója szemből nézve

Az egyes tűk összeköttetései a következők:

1. **A** csatorna
2. **B** csatorna
3. **C** csatorna
4. **D** csatorna
5. Digitális bemenet/kimenet 0
6. Digitális bemenet/kimenet 2
7. Föld (GND)
8. +5 V
9. Föld (GND)
10. Föld (GND)
11. DAC_1
12. DAC_2
13. Digitális bemenet/kimenet 1
14. Digitális bemenet/kimenet 3
15. −5 V

3. A kommunikáció sémája

Az eszköznek két üzemmódja van, *folyamatos* és *nemfolyamatos*. Alapértelmezett a nemfolyamatos üzemmód, folyamatosba az `@S` paranccsal vihető át, és mindaddig folyamatos üzemmódban marad, amíg meg nem szakítjuk egy `escape` karakter, azaz egy [27] értékű bájt leküldésével.

Az alapértelmezett nemfolyamatos üzemmódban az eszköz minden leküldött bájtot visszaküld, így ad visszajelzést a bájt fogadásáról. A parancsokat is bájtonként kell leküldeni, a bájtot visszavárni, és csak akkor haladhatunk tovább a következő bájtra, ha a visszakapott bájt megegyezik az elküldött bájttal, különben kommunikációs hibára gyanakodhatunk (pl folyamatos üzemmódban vagyunk, vagy egy előző parancs után nem olvastuk ki az eszköz válaszának minden bájtját). Az így megvalósított kommunikációt a 2. példa szemlélteti `C#` nyelven.

Az eszköz minden elküldött bájtot visszaküld, így meggyőződhetünk egyszerűen arról, hogy fizikailag működik-e a kommunikáció a számítógép és az eszköz között. Ez alól kivétel a folyamatos üzemmód és az `@I` parancs.

4. Folyamatos üzemmód

A folyamatos üzemmód az `@S` paranccsal indítható és az `escape` karakter – [27] – leküldésével állítható meg. Előzőleg a `@c` paranccsal adhatjuk meg, hogy melyik analóg-digitális átalakítóhoz melyik csatornát rendelje az eszköz és milyen erősítéssel, az `@f` parancs pedig a mintavételi frekvenciát állítja be. *Folyamatos üzemmódban az eszköz nem küldi vissza a leküldött bájtokat.*

Az `@S` parancs leküldése után megindul a mintavételezés az előzőleg az `@f` paranccsal beállított mintavételi frekvenciával, és az eszköz elkezdi küldeni fölfelé az adatokat. Az adatsor 4 adat hosszúságú blokkokból áll a következők szerint: $\langle ADC_1$ *adata* t *időpillanatban az 1. csatornán*$\rangle$-$\langle ADC_2$ *adata* t *időpillanatban a 2. csatornán*$\rangle$-$\langle ADC_1$ *adata* $t + \Delta t$ *időpillanatban a 3. csatornán*$\rangle$-$\langle ADC_2$ *adata* $t + \Delta t$ *időpillanatban a 4. csatornán*$\rangle$, ahol $\Delta t = 1/f_{\mathrm{m}}$ a mintavételi időköz (f_{m} a mintavételi frekvencia). Egy adat két bájtból állt, tehát a blokkok 8 bájt hosszúak.

Az adatok küldése *nem folytonos,* hanem az eszköz összevár alapértelmezésben 128 adatot (azaz 256 bájtot), amit egyben küld el, utána ismét összevár 128 adatot, stb. A küldési puffer mérete, azaz az összevárt adatok száma a `@b` paranccsal állítható 1 és 255 között. Ha kisebb mintavételezési frekvenciával dolgozunk, érdemes a pufferméretet csökkenteni, különben a megjelenítés szaggatottnak fog tűnni (pl 128 Hz mintavételi frekvenciánál alapértelmezésben csak másodpercenként jutunk hozzá az új adatokhoz).

5. Az adatok feszültséggé konvertálása

Az eszközben lévő analóg-digitális átalakítók 16-bitesek, így egy-egy adatot két bájt reprezentál. Jelölje az ugyanazon adathoz tartozó két bájtot $[b_1]\,[b_0]$ az adatsorban. A kommunikáció logikája *big-endian,* azaz $[b_1]$ a nagyobb helyiértékű bájt.

A két bájtból először előállítunk egy 16-bites (vagy nagyobb) egész számot (z), például úgy, hogy $[b_1]$ értékét balra toljuk 8 bittel és az eredményhez hozzáadjuk $[b_0]$-t (lásd az 1. példát). Ha megvan az egész számunk, azt kell figyelembe vennünk, hogy a konverter a $-5\,\mathrm{V}$ és $+5\,\mathrm{V}$ közti tartományt skálázta be $2^{16} = 65536$ részre:

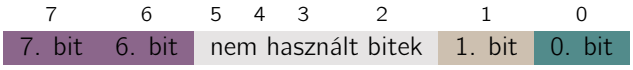
$$\Delta U = \frac{+5\,\mathrm{V} - (-5\,\mathrm{V})}{2^{16}} = \frac{10}{65536}\,\mathrm{V} = \frac{5}{32768}\,\mathrm{V}.$$

Ha a z digitalizált számból akarjuk visszakapni az U feszültséget, a legkisebb feszültséghez kell hozzáadni a ΔU feszültségkvantum z-szeresét:

$$U = U_{\mathrm{min}} + z \cdot \Delta U = -5\,\mathrm{V} + z \cdot \frac{5}{32768}\,\mathrm{V} = 5\,\mathrm{V} \cdot \left(\frac{z}{32768} - 1\right). \tag{1}$$

6. Jelmagyarázat

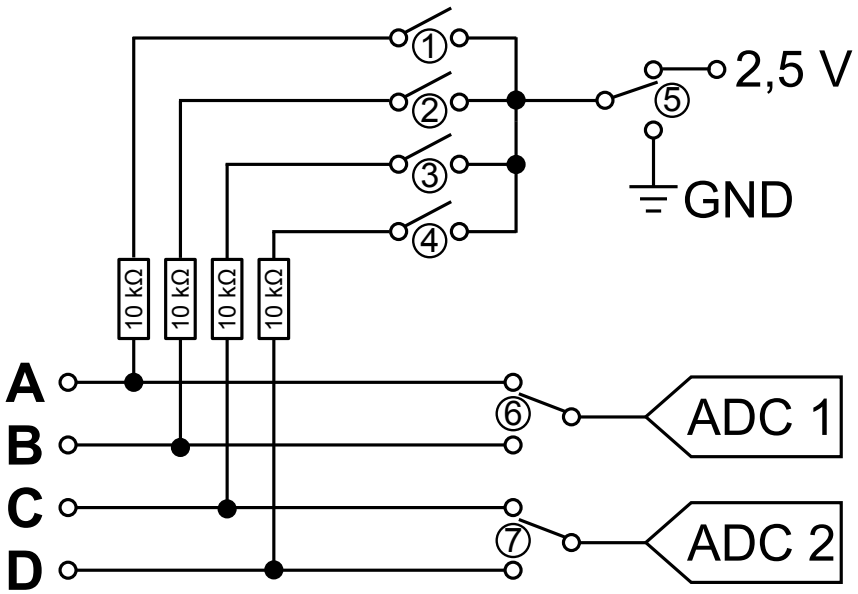
- `[b]`: egyetlen *b* bájt
- `@p[b1][b0]`: `@p` parancs `[b1][b0]` argumentumokkal
- a bájt belső szerkezetét ábrázoló bittérkép (a különböző funkciójú bitek más színekkel):



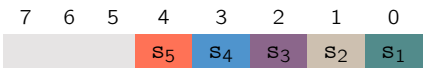
7. Parancsok

`@I` – **szöveges eszközinformáció kérése.** A parancs kiadása után az eszköz egy karakter leküldésére az eszközinformáció egy karakterét küldi válaszul. Célszerű olyan vezérlőkaraktert választani, ami az eszközinformációban biztosan nem szerepelhet. Amikor először ezt a karaktert kapjuk vissza, tudhatjuk, hogy az eszközinformációt tartalmazó karakterlánc végére értünk, hiszen a leküldött karakter visszaküldése megfelel az alapüzemmódnak. [3. példa]

`@x[b0]` – **ellenállások kapcsolása a bemenet és a tápfeszültség/föld közé.** Ahogy a 2. ábrán láthatjuk, a bemenetek nem szükségképpen kerülnek közvetlenül az analóg-digitális átalakítóra (pontosabban az az előtt lévő előerősítő bemenetére), hanem lehetőség van arra, hogy feszültségosztós kapcsolásokat valósítsunk meg a bemeneten. Ez például termisztoros kapcsolásoknál hasznos, hiszen ekkor elegendő a termisztort a föld és a bemenet közé kötni, nem kell ellenállásosztóba kötni, az ellenállásosztó a műszeren belül megvalósítható (például az ① és a ⑤ kapcsoló együttes bekapcsolásával). A kapcsolók állását az `@x` parancs argumentumaként megadott `[b0]` bájt szabályozza a következők szerint:



2. ábra. A bemeneten lévő vezérelhető kapcsolók elrendezése



ahol a bitek jelentése a következő:

- 0. bit (s_1): az ① kapcsoló állása. Ha 0, a kapcsoló nyitott, ha 1, a kapcsoló zárt.
- 1. bit (s_2): a ② kapcsoló állása. Ha 0, a kapcsoló nyitott, ha 1, a kapcsoló zárt.
- 2. bit (s_3): a ③ kapcsoló állása. Ha 0, a kapcsoló nyitott, ha 1, a kapcsoló zárt.

- 3. bit (s_4): a ④ kapcsoló állása. Ha 0, a kapcsoló nyitott, ha 1, a kapcsoló zárt.
- 4. bit (s_5): a ⑤ kapcsoló állása. 0 értéke esetén 0 V van az ellenállásosztó végére kapcsolva, 1 értéke esetén pedig 2,5 V.
- 5–7. bit: *nincs jelentése*

A parancs használatára a 4. kódrészletben láthatunk példát.

`@t[b0]` **– a trigger invertálása.** Invertálja a triggert, ha a $[b_0]$ bájt értéke 0-tól különböző.

`@S` **– a folyamatos üzemmód elindítása.** A parancs hatására elindul a folyamatos üzemmód. Az `@f` paranccsal megadott mintavételi frekvenciával megindul a mintavételezés a `@c` paranccsal előzőleg megadott bemeneteken, az ott beállított erősítéssel. A két analóg-digitális átalakító, ADC_1 és ADC_2, szimultán mintavételezi a `@c` paranccsal beállított első két csatornát, majd a következő órajelütemben rátér a `@c` paranccsal beállított második két csatornára (ez természetesen lehet ugyanaz is, mint az első kettő).

Az eszköz annyi adatot vár össze, amennyit a `@b` paranccsal beállítottunk (alapértelmezésben 128), mielőtt elküldené őket. Egy adat két bájt hosszúságú. Az adatfolyam 4 adat (azaz 8 bájt) hosszúságú blokkokra tagolódik az alábbiak szerint:

$\langle ADC_1$ *adata* t *időpillanatban az 1. csatornán*$\rangle$-$\langle ADC_2$ *adata* t *időpillanatban a 2. csatornán*$\rangle$-$\langle ADC_1$ *adata* $t + \Delta t$ *időpillanatban a 3. csatornán*$\rangle$-$\langle ADC_2$ *adata* $t + \Delta t$ *időpillanatban a 4. csatornán*$\rangle$,

ahol $\Delta t = 1/f_{\mathrm{m}}$ a mintavételi időköz (f_{m} a mintavételi frekvencia).

Folyamatos üzemmódban az eszköz nem küldi vissza a leküldött bájtokat. Folyamatos üzemmódból az `escape` karakter, azaz egy [27] értékű bájt leküldésével lehet kilépni.

`@s[f2][f1][f0][n1][n0]` **– egyszeri adatsor mérése.** A parancs hatására az aktív analóg-digitális átalakítón n darab adatot mér az eszköz f mintavételi frekvenciával, majd a teljes adatsort fölküldi a számítógépre. Ha mindkét átalakító aktív, a mérési adatok fölváltva követik egymást.

- $[n_1][n_0]$: az adatok számának bájtjai (a nagyobb helyiértékű van elöl)
- $[f_2][f_1][f_0]$: a mintavételi frekvencia bájtjai (a nagyobb helyiértékű van elöl)

`@A[b0]` **– az aktív analóg-digitális átalakító kiválasztása.** Az argumentumban megadott bájt dönti el, hogy melyik analóg-digitális átalakító aktív:

- 0. bit (a_1): az ADC_1 ki- vagy bekapcsolásának bitje (`1`: bekapcsolva)
- 1. bit (a_2): az ADC_2 ki- vagy bekapcsolásának bitje (`1`: bekapcsolva)
- 2–7. bit: *nincs jelentése*

Értelemszerűen, ha $[b_0] = 3$, akkor mindkét analóg-digitális átalakítót bekapcsoljuk.

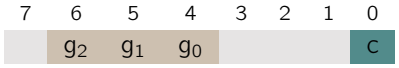
`@1[b0]` **– az ACD_1 analóg-digitális átalakítóhoz rendelt csatorna és az átalakítás előtti erősítés beállítása.** Az argumentumként megadott bájt szabja meg, hogy ADC_1 melyik csatornát mérje és mekkora erősítéssel. Az erősítés értéke 2^g, ahol g értékét $[b_0]$ 4–6. bitje tartalmazza.

- 0. bit (c): ha értéke 0, az **A** csatorna lesz az ADC_1-re kapcsolva; különben a **B** csatorna (lásd a ⑥-os kapcsolót a 2. ábrán)
- 1–3. bit: *nincs jelentése*
- 4–6. bit: az erősítést megszabó g szám bitjei. Erősítés $= 2^g$.

- 7. bit: *nincs jelentése*

A parancsnak folyamatos üzemmódban nincs hatása. Folyamatos üzemmódban a csatorna-hozzárendeléseket és az erősítéseket a @c parancs határozza meg.

@2[b_0] – az ACD$_2$ analóg-digitális átalakítóhoz rendelt csatorna és az átalakítás előtti erősítés beállítása. Az argumentumként megadott bájt szabja meg, hogy ADC$_2$ melyik csatornát mérje és mekkora erősítéssel. Az erősítés értéke 2^g, ahol g értékét [b_0] 4–6. bitje tartalmazza.



- 0. bit (c): ha értéke 0, az **C** csatorna lesz az ADC$_2$-re kapcsolva; különben a **D** csatorna (lásd a ⑦-es kapcsolót a 2. ábrán)
- 1–3. bit: *nincs jelentése*
- 4–6. bit: az erősítést megszabó g szám bitjei. Erősítés $= 2^g$.
- 7. bit: *nincs jelentése*

A parancsnak folyamatos üzemmódban nincs hatása. Folyamatos üzemmódban a csatorna-hozzárendeléseket és az erősítéseket a @c parancs határozza meg.

@f[b_1][b_0] – mintavételi frekvencia megadása folyamatos üzemmódban. A paranccsal beállíthatjuk a mintavételi frekvenciát az alábbiak szerint:

$$f_m = b_1 \cdot 256 + b_0,$$

azaz [b_1] a mintavételi frekvencia fölső bájtját, [b_0] pedig az alsót adja meg. A parancsnak nincs hatása az egyszeri adatsor mérése során alkalmazott mintavételi frekvenciára, azt a @s parancs maga állítja be.

@M[b_0] – egyszeri mérés b_0 átlagolással. A parancs hatására b_0 számú mérést végez az eszköz mindkét analóg-digitális átalakítón, az eredményeket átlagolja és fölküldi a számítógépre. A válasz négy bájt, amelyből az első kettő az ADC$_1$ adata (a nagyobb helyiértékű bájttal kezdve), a másik kettő az ADC$_2$ adata. Ha csak az egyik analóg-digitális átalakító aktív, akkor a két-két bájt azonos, és értelemszerűen az aktív átalakító mérési eredményét tartalmazza.

@d[b_1][b_0] – az első digitális-analóg átalakító, DAC$_1$, kimenetének beállítása. A parancs argumentumának két bájtjával azt a z kódot kell megadnunk, ami a kívánt U feszültségnek megfelel. A kód előállításához az (1) egyenletet kell megfordítanunk, figyelembe véve, hogy a digitális-analóg konverter 12-bites, azaz $2^{12} = 4096$ lehetőség kódolására alkalmas:

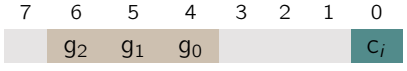
$$z = 2048 \cdot \left(\frac{U}{5\,\mathrm{V}} + 1 \right).$$

Az argumentum bájtjai a z szám fölső és alsó bájtját tartalmazzák:

$$z = b_1 \cdot 256 + b_0.$$

@D[b_1][b_0] – a második digitális-analóg átalakító, DAC$_2$, kimenetének beállítása. Lásd a @d parancs leírásánál.

@c[b_1][b_2][b_3][b_4] – a folyamatos üzemmód csatorna-hozzárendeléseinek és erősítéseinek beállítása. A folyamatos üzemmód 4 csatorna mérésére alkalmas (ebből két csatornát mér egyszerre, a másik két csatorna mérése egy órajelütemmel eltolva történik). A @c parancs argumentumában megadott 4 bájt a 4 mérendő csatorna kiválasztásért és az erősítés beállításáért felel, ugyanolyan szintaxissal, mint az @1 és @2 parancsok esetében. A [b_i] bájt az i-edik csatornára vonatkozik, szerkezete pedig a következő:



- 0. bit (c_i): ha értéke 0, az első lehetséges csatorna (ADC_1 esetében az **A** csatorna, ADC_2 esetében a **C** csatorna) lesz az ADC_i-re kapcsolva; különben a második lehetséges csatorna (ADC_1 esetében a **B** csatorna, ADC_2 esetében a **D** csatorna)
- 1–3. bit: *nincs jelentése*
- 4–6. bit: az erősítést megszabó g szám bitjei. Erősítés $= 2^g$.
- 7. bit: *nincs jelentése*

`@b[n0]` **– adatküldési puffer méretének megadása.** Az eszköz bizonyos számú adatot összevár, mielőtt fölküldené a számítógépnek. Ez az érték alapértelmezésben 128, de a `@b` paranccsal módosítható. A parancs `[n0]` argumentuma megadja, hogy mekkora legyen a küldési adatblokk hosszúsága. Értéke adatban értendő, azaz az adatblokk bájtjainak száma $2 \cdot n_0$.

`@w` **– a jelgenerátor kikapcsolása.** Kikapcsolja a jelgenerátort.

`@W` **– a jelgenerátor bekapcsolása.** Bekapcsolja a `@L` paranccsal beállított paraméterek mellett a jelgenerátort.

`@L[d][a][e]` **– a jelgenerátor beállítása.** Jelgenerátorként a DAC_1 szolgál. A jelgenerátor minden órajelütemben változtatja a DAC_1-re kiadott értéket. Az értékeket egy 256 pontból álló szinusztáblázatból veszi, ezt lehet módosítani az `@L` parancs argumentumában megadott értékekkel:

- `[d]`: megadja, hogy a generátor hányasával lép a táblázatban. Ezzel a kiadandó szinusz frekvenciája változtatható az f_m mintavételi frekvenciához viszonyítva. Értelemszerűen csak sokszorozni lehet a frekvenciát.
- `[a]`: az amplitúdót állítja be
- `[e]`: az egyenfeszültségű szintet (az offszetet) szabályozza

8. Példák C# nyelven

```
public const int WordLength = 2;    // The number of bytes per data point.

protected double[] ConvertToVoltages(byte[] bytes)
{
   double[] voltages = new double[bytes.Length / WordLength];
   int v;   // Voltage as 16-bit code.

   for (int i = 0; i < voltages.Length; i++) {
      v = bytes[WordLength * i] << 8;  // TODO: This code assumes 2-byte words, will not
          work for other word lengths.
      v += bytes[WordLength * i + 1];
      voltages[i] = 5.0 * (v / 32768.0  - 1.0); // 16 bits between -5 V and +5 V; this is
          [10.0 * v / 65536.0 - 5.0] simplified.
   }

   return voltages;
}
```

1. példa. A digitalizált adat feszültséggé konvertálása

```
protected void Send(byte[] bytes)
{
   byte[] read;
   foreach (byte b in bytes) {
      Port.Write(new byte[] { b });
      read = Port.Read(1);
      if (read[0] != b) {
         Exception exception = new IOException("Sent and received bytes do not match.");
         exception.Data.Add("Sent byte", b);
         exception.Data.Add("Received byte", read[0]);
```

```
            throw exception;
        }
    }
}
```

2. példa. Adatok küldése

```
public const string DeviceInfoCommand = "@I";

public string GetDeviceInfo()
{
    // [...]
    string deviceInfo = String.Empty;
    byte[] commandBytes = ConvertToBytes(DeviceInfoCommand, null);
    Encoding ascii = Encoding.ASCII;
    bool continueReading = true;
    string command = "§";
    byte[] cb = ConvertToBytes(command, null);
    byte[] read;
    //
    Send(commandBytes);
    while (continueReading) {
        Port.Write(cb);
        read = Port.Read(1);
        continueReading = (read[0] != cb[0]);
        if (continueReading) {
            deviceInfo += ascii.GetString(read);
        }
    }
    // [...]
    return deviceInfo;
}
```

3. példa. Eszközinformáció kérése

```
[Flags]
public enum ResistanceSwitching : byte {
    All = 15,
    None = 0,
    ChannelA = 1,
    ChannelB = 2,
    ChannelC = 4,
    ChannelD = 8,
    SupplyVoltage = 16,
    NotDefined = 32
}

public const string ResistanceSwitchingCommand = "@x";

public void SetResistanceSwitching(ResistanceSwitching switching)
{
    // [...]
    byte[] commandbytes = ConvertToBytes(ResistanceSwitchingCommand, new byte[] {
        (byte)switching });

    try {
        Send(commandbytes);
        resistanceSwitching = switching;
    } catch (Exception exception) {
        resistanceSwitching = ResistanceSwitching.NotDefined;
        throw exception;
    }
    // [...]
}
```

4. példa. Az ellenállások kapcsolása