

Texas Instruments MSP-EXP430FR5739 kártya

Gyűjtsük ki a szükséges és hasznos Texas adatlapokat, user manualokat, workshop dokumentumokat (dokumentum pontos neve és pontos letöltési link formában), szóval állítsuk össze az "irodalomjegyzéket"!

A Texas Instruments **MSP-EXP430FR5739** (ismertebb nevén a "Fraunchpad") egy különleges fejlesztői kártya, mivel ez vezette be az **FRAM** (Ferroelectric Random Access Memory) technológiát az MSP430-as családba.

Az alábbiakban összegyűjtöttem a fejlesztéshez elengedhetetlen hivatalos dokumentációkat és segédleteket:

1. Alapvető Hardver Dokumentáció (Eszközspecifikus)

Ezek a dokumentumok a konkrét mikrokontroller (MCU) fizikai és logikai felépítését írják le.

- **MSP430FR5739 Mixed-Signal Microcontrollers Data Sheet**
 - *Leírás:* Tartalmazza a lábkiosztást (pinout), az elektromos paramétereket és a memória térképet.
 - [Letöltési link](#)
 - **MSP430FR57xx Family User's Guide**
 - *Leírás:* A regiszterek szintű leírása. Minden periféria (órajel, ADC, UART, SPI, Timer) részletes működését itt találod.
 - [Letöltési link](#)
-

2. A Fejlesztői Kártya (LaunchPad) Dokumentációja

Ezek a dokumentumok magáról a piros kártyáról, a ráépített gyorsulásmérőről, LED-ekről és a beépített programozóról szólnak.

- **MSP-EXP430FR5739 Experimenter Board User's Guide**
 - *Leírás:* A kártya kapcsolási rajza, a demo szoftver leírása és a jumpelek funkciói.
 - [Letöltési link](#)
 - **MSP-EXP430FR5739 Hardware Design Files**
 - *Leírás:* Eagle vagy Altium tervezőknek: a NYÁK terveket tartalmazó csomag (zip).
 - [Letöltési link](#)
-

3. Szoftveres Segédletek és Könyvtárak

A modern fejlesztéshez a regiszterek közvetlen bizergálása helyett érdemes a gyári library-eket használni.

- **MSP430 DriverLib for MSP430FR5xx_6xx Devices**
 - *Leírás:* API dokumentáció a DriverLib-hez, ami absztrakciós réteget biztosít a perifériák kezeléséhez.
 - [Letöltési link](#)
 - **MSP430Ware (Software Resource Explorer)**
 - *Leírás:* Ez egy szoftvercsomag, ami tartalmazza az összes kódpéldát (C és Assembly).
 - [Letöltési link \(Webes elérés\)](#)
-

4. Workshopok és Oktatási Anyagok

A Texas Instruments korábban nagyon részletes workshopokat készített, amelyek lépésről lépésre tanítanak meg az FRAM használatára.

- **MSP430 FR57xx Workshop - Student Guide**
 - *Leírás:* Egy teljes tananyag, ami elmagyarázza az FRAM előnyeit és laborfeladatokon vezet végig.
 - [Letöltési link](#)
- **MSP430 FRAM Quality and Reliability (Application Report)**
 - *Leírás:* Ha érdekli a technológiai háttér, ez a dokumentum taglalja az FRAM élettartamát és írási ciklusait.
 - [Letöltési link](#)

Hasznos tipp a kezdéshez:

A professzionális kódfejlesztéshez a **Code Composer Studio (CCS)** vagy az **IAR Embedded Workbench** használata javasolt. Ha a parancssori megoldások híve vagy, az **MSP430-GCC** fordítót érdemes letölteni. [Kezdőknek az ENERGIA ajánlható.](#)

Az összeállított irodalomjegyzékből a **User's Guide (SLAU272)** a legfontosabb darab. Ha a regiszterek szintjén akarsz belemenni az FRAM vezérlésébe vagy a memóriavédelembe (MPU), az lesz a bibliád.

Van esetleg olyan konkrét periféria ezen a kártyán, aminek a programozásához keresel még célzottabb példát vagy Application Note-ot?

Van itt még egy gyorsulásmérő, ahhoz is kellene adatlap.

Természetesen, a gyorsulásmérőhöz tartozó adatok is fontosak, hiszen ez az egyik fő szenzor a kártyán.

Az **MSP-EXP430FR5739** kártyára egy **Analog Devices ADXL335** típusú, 3-tengelyes analóg gyorsulásmérő van integrálva. Ez nem digitális (I2C/SPI) interfészen kommunikál, hanem a három tengely (X, Y, Z) gyorsulásával arányos feszültséget ad ki, amit a mikrokontroller ADC12 (analóg-digitális konverter) moduljával kell beolvasnod.

Itt az irodalomjegyzék kiegészítése:

5. A gyorsulásmérő dokumentációja

- **ADXL335: Small, Low Power, 3-Axis ± 3 g Accelerometer Data Sheet**
 - *Leírás:* A szenzor elektromos jellemzői, érzékenysége (300 mV/g) és a sávszélesség beállításához szükséges kondenzátorok méretezése.
 - [Letöltési link](#)
-

6. Bekötési segédlet (Board Mapping)

A kód írásakor az alábbi lábkiosztásra lesz szükséged a P3 porton:

Tengely	MCU Láb (Analóg bemenet)	Regiszter beállítás
X tengely	P3.0 / A12	ADC12INCH_12
Y tengely	P3.1 / A13	ADC12INCH_13
Z tengely	P3.2 / A14	ADC12INCH_14

Fontos szakmai apróság: A kártyán van egy engedélyező láb is a gyorsulásmérőhöz (**ACC_PWR**), ami a **P2.7**-es lábra van kötve. Ezt szoftverből HIGH szintre kell húznod ahhoz, hogy a szenzor tápfeszültséget kapjon és mérni kezdjen. Ha ez elmarad, csak zajt fogsz látni az ADC csatornákon.

Ez így már kerek a "kötelező irodalomhoz", vagy maradt még kérdéses alkatrész a kártyán? (Van még rajta egy NTC termisztor is a hőmérséklethez, ha az is érdekes.)

Igen, a termisztorról is kellene információ, illetve van egy LDR fényérzékeny ellenállás is

Valóban, a kártya bal felső sarkában ott figyelnek az analóg szenzorok is. Ezek a legegyszerűbb alkatrészek, de a pontos méréshez ismerni kell a karakterisztikájukat.

Íme a kiegészítés az "irodalomjegyzékhez":

7. Környezeti érzékelők (NTC és LDR)

- **NTC Termisztor (Hőmérséklet mérés)**
 - *Típus:* **Murata NCP18WF104J03RB** (vagy hasonló 100 k Ω -os NTC).

- *Leírás:* 100 kΩ-os ellenállás 25°C-on. A kártyán egy feszültségosztó része, aminek a kimenetét az ADC méri.
 - [Adatlap letöltése](#)
 - **LDR (Fényérzékelő ellenállás)**
 - *Típus:* Általános CdS (Kadmium-szulfid) fotocellás ellenállás.
 - *Leírás:* A megvilágítással fordítottan arányos az ellenállása (sötétben nagy, fényben kicsi). Szintén egy feszültségosztóba van kötve.
 - [Általános LDR karakterisztika \(Példa\)](#)
-

8. Hardveres bekötés a szoftverhez

Ahhoz, hogy az irodalmat kódra tudd váltani, ezeket a lábakat kell konfigurálnod:

Szenzor	MCU Láb (Analog bemenet)	ADC12 Csatorna	Megjegyzés
NTC Termisztor	P3.5 / A5	ADC12INCH_5	Feszültségosztó felső tagja.
LDR (Fény)	P3.4 / A4	ADC12INCH_4	Minél több a fény, annál magasabb az ADC érték.

Pro tipp az oktatáshoz: Mivel ezek passzív szenzorok egy feszültségosztóban, itt is szükség van a tápfeszültség kapcsolására. A **P2.7 (ACC_PWR)** lábat itt is HIGH szintre kell húzni, mert ez a vonal táplálja a gyorsulásmérőt, a termisztort és a fényérzékelőt is. Ha spórolni akarsz az energiával, mérés után ezt lekapcsolhatod.

Ez így már egy teljes "full-stack" lista a kártyához. Van még valami, amit rátennénk a listára, vagy jöhet a regiszter-szintű konfigurálás?

A következő lépés az ENERGIA ismertetése (ezzel Arduino nyelven programozható a kártya. Erről is kérem begyűjteni a releváns információt.

Az **Energia** egy nyílt forráskódú, közösségi fejlesztésű IDE és szoftverkeretrendszer, amely az Arduino ökoszisztémát ülteti át a Texas Instruments mikrovezérlőire. Az MSP-EXP430FR5739 kártya az egyik első támogatott platform volt, így kiválóan kezelhető ezzel az eszközzel.

Íme az irodalomjegyzék és a technikai részletek az Energia használatához:

9. Energia Szoftver és Dokumentáció

- **Energia Hivatalos Weboldal & Letöltés**
 - *Leírás:* A fejlesztőkörnyezet letöltése és a támogatott kártyák listája.
 - [Link az oldalhoz](#)
- **Energia Guide: MSP-EXP430FR5739 Experimenter Board**

- *Leírás:* Specifikus útmutató ehhez a kártyához, a lábkiosztás és a hardveres sajátosságok Arduino-szemponitú megközelítésével.
 - [Link az útmutatóhoz](#)
 - **Energia GitHub Repository (Hardware Core)**
 - *Leírás:* Itt található a forráskód az msp430 maghoz, ha bele akarsz nézni a regiszter-absztrakciókba.
 - [GitHub Link](#)
-

10. A PIB (Pin Information Block) Diagram

A PIB diagram az Energia legfontosabb "térképe". Ez mondja meg, hogy az Arduino-stílusú lábszámok (pl. P1_0, D1, A0) melyik fizikai lábnak felelnek meg a Fraunchpad kártyán.

A Fraunchpad PIB főbb jellemzői:

Funkció	Energia láb megnevezése	Fizikai láb (MCU)
LED 1..8	P3_4, P3_5, P3_6, P3_7, P1_0, P1_1, P1_2, P1_3	A kártyán lévő 8 piros LED sora.
S1 Nyomógomb	P4_0	Felhúzó ellenállás szükséges vagy INPUT_PULLUP.
S2 Nyomógomb	P4_1	Felhúzó ellenállás szükséges vagy INPUT_PULLUP.
UART (Serial)	P2_0 (TX), P2_1 (RX)	Ez a csatorna megy át az USB-illesztőn a PC-re.
I2C (Wire)	P1_6 (SDA), P1_7 (SCL)	Külső szenzorokhoz.

11. Fontos tudnivaló az FRAM és az Energia kapcsolatáról

Az Energia alapról nem használja ki az FRAM minden különleges képességét (pl. nem kezeli automatikusan a nem-felejtő változókat), de az alábbi kiegészítő könyvtárral ez elérhető:

- **MSP430 FRAM Utilities for Energia**
 - *Leírás:* Lehetővé teszi változók tárolását az FRAM-ben a `__attribute__((section(".upper.rodata")))` vagy hasonló direktívák nélkül, Arduino stílusban.
 - [Letöltési link/Dokumentáció](#)

Technikai megjegyzés:

A kártyán lévő analóg szenzorok (gyorsulásmérő, termisztor, LDR) eléréséhez az Energia `analogRead()` függvényét használhatod, de ne felejtse el a `pinMode(P2_7, OUTPUT);` `digitalWrite(P2_7, HIGH);` parancsokat a `setup()`-ban, különben a szenzorok nem kapnak tápfeszültséget!

Energia használata esetén kell driver a programozóhoz, vagy a Windows felismeri?

Ez egy fontos gyakorlati kérdés, mert a **Fraunchpad** (MSP-EXP430FR5739) egy régebbi fejlesztői kártya, és a rajta lévő **eZ430 lite** emulátor/programozó interfész nem mindig "plug-and-play" a modern Windows rendszereken.

Íme a pontos helyzet és a megoldás:

1. Kell-e driver?

Igen, szükség van specifikus driverre. Bár a Windows megpróbálhatja felismerni, az esetek többségében "MSP430-UART" vagy "Unknown Device" néven jelenik meg a szoftveres illesztő nélkül. Az **Energia** telepítője elvileg tartalmazza ezeket, de sokszor manuálisan kell őket telepíteni vagy frissíteni.

2. Honnan töltsd le?

A legbiztosabb forrás a Texas Instruments hivatalos driver csomagja, amely tartalmazza az összes LaunchPad és Experimenter kártyához szükséges USB-illesztőt:

- **MSP430 USB Debug Interface Drivers**
 - *Pontos név:* MSP430-Drivers-1.0.1.1-Setup.exe (vagy frissebb verzió)
 - [Letöltési link](#)

3. Mit kell látnod az Eszközkezelőben?

Ha a telepítés sikeres és rádugod a kártyát, a Windows Eszközkezelőben (**Device Manager**) az alábbi két bejegyzést kell keresned a **Ports (COM & LPT)** szekció alatt:

1. **MSP430 Application UART (COMx)** – Ezen keresztül megy a soros terminál (Serial monitor).
2. **MSP Debug Interface (COMy)** – Ezen keresztül tölti fel az Energia a kódot.

4. Gyakori hiba az Energiában

Ha a driver fent van, de az Energia mégis hibát dob feltöltéskor:

- **Eszköz kiválasztása:** Menj az *Eszközök (Tools)* -> *Alaplap (Board)* menübe és válaszd ki pontosan: **MSP430FR5739**.
- **Programozó:** Az *Eszközök (Tools)* -> *Programozó (Programmer)* menüben általában az **MSP430-GCC** vagy a **FET430UIF** beállítással működik (a Fraunchpad az emulált FET-et használja).

Ha telepítetted a drivert és mégsem látod a COM portokat, érdemes megpróbálni egy másik USB kábelt, mert a régi Fraunchpad kártyák néha kényesek a kábel minőségére!

Írjunk néhány egyszerű és didaktikus mintaprogramot ENERGIA-ban, kommentekben magyarázva, hogy mit, hogyan, miért csinálunk!

- LED villogtatás
- Futófény a LED soron
- Hőmérés az NTC-vel
- Fénymérés az LDR-rel
- A gyorsulásmérő kiolvasása és kiírás a soros porton (ARDUINO IDE Plotterrel kompatibilis módon - nem tudom, hogy az Energiában van-e ilyen lehetőség?)
- Egyszerű zenelejátszó passzív buzzer felhasználásával.

Az Energia IDE remek választás ezekhez a feladatokhoz, mert szinte az összes Arduino-függvény (digitalWrite, analogRead, Serial.print) ugyanúgy működik. Az Energia tartalmaz **Serial Plottert** is (az Eszközök menüben találod), ami automatikusan kirajzolja a számértékeket, ha azok vesszővel elválasztva érkeznek.

Íme a didaktikus mintapéldák:

1. LED villogtatás (A "Hello World")

Ez a legegyszerűbb program, ami megtanítja a lábak kimenetként való definiálását.

```
// A Fraunchpad kártyán a RED_LED a P1_0 lábra van kötve alapértelmezés szerint
void setup() {
  pinMode(RED_LED, OUTPUT); // Beállítjuk a lábat kimenetnek
}

void loop() {
  digitalWrite(RED_LED, HIGH); // Bekapcsoljuk (magas szint)
  delay(500);                  // Várunk 500 ezredmásodpercet
  digitalWrite(RED_LED, LOW);  // Kikapcsoljuk (alacsony szint)
  delay(500);                  // Ismét várunk
}
```

2. Futófény a 8 bites LED-soron

A Fraunchpad különlegessége a 8 sorba rendezett LED. Itt egy tömböt használunk a lábak kezelésére.

```
// Definiáljuk a LED-ek lábait egy tömbben a PIB alapján
int leds[] = {P1_0, P1_1, P1_2, P1_3, P3_4, P3_5, P3_6, P3_7};

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(leds[i], OUTPUT); // Minden LED lábat kimenetre állítunk
  }
}

void loop() {
  for (int i = 0; i < 8; i++) {
    digitalWrite(leds[i], HIGH); // Aktuális LED be
    delay(100);                  // Rövid várakozás
    digitalWrite(leds[i], LOW);  // Aktuális LED ki
  }
}
```

3. Hőmérés (NTC) és Fénymérés (LDR)

A két analóg szenzort egy kalap alá vesszük. **Kritikus:** a P2_7 lábat (ACC_PWR) magasra kell húzni, mert ez ad tápot a szenzoroknak!

```
void setup() {
  Serial.begin(9600);           // Soros port indítása
  pinMode(P2_7, OUTPUT);       // A szenzorok tápvonalának beállítása
  digitalWrite(P2_7, HIGH);    // Tápfeszültség bekapcsolása (Fontos!)
}

void loop() {
  int feny = analogRead(P3_4); // LDR olvasása (A4 csatorna)
  int ho = analogRead(P3_5);   // NTC olvasása (A5 csatorna)

  Serial.print("Feny: ");
  Serial.print(feny);
  Serial.print(" | Homerséklet (nyers): ");
  Serial.println(ho);

  delay(1000); // Másodpercenként frissítünk
}
```

4. Gyorsulásmérő és Serial Plotter

A Serial Plotter akkor rajzol szép görbéket, ha az értékeket szóközzel vagy vesszővel elválasztva küldjük, a sor végén pedig sortörés van.

```
void setup() {
  Serial.begin(9600);
  pinMode(P2_7, OUTPUT);
  digitalWrite(P2_7, HIGH); // Gyorsulásmérő tápellátása
}

void loop() {
  // A tengelyek kiolvasása a specifikus lábokról
  int x = analogRead(P3_0);
  int y = analogRead(P3_1);
  int z = analogRead(P3_2);

  // Plotter formátum: érték1, érték2, érték3
  Serial.print(x);
  Serial.print(",");
  Serial.print(y);
  Serial.print(",");
  Serial.println(z); // Itt van a lezáró sortörés

  delay(20); // 50Hz-es mintavételezés a sima görbékért
}
```

5. Egyszerű zenelejátszó (Passzív buzzer)

Mivel a kártyán nincs beépített buzzer, egy passzív piezo hangszórót kell kötnöd például a P1_4 lábra és a GND-re. A `tone()` függvényt használjuk.

```
#define BUZZER_PIN P1_4

// Dallam: C4, G3, G3, A3, G3, 0, B3, C4
int dallam[] = {262, 196, 196, 220, 196, 0, 247, 262};
int idotartam[] = {4, 8, 8, 4, 4, 4, 4, 4}; // Negyed, nyolcad stb.

void setup() {
  // A setupban játszunk le egyszer
  for (int i = 0; i < 8; i++) {
    int hossza = 1000 / idotartam[i];
    if (dallam[i] > 0) {
      tone(BUZZER_PIN, dallam[i], hossza);
    } else {
      delay(hossza);
    }
    // Szünet a hangok között, hogy elkülönüljenek
    int szunet = hossza * 1.30;
    delay(szunet);
    noTone(BUZZER_PIN);
  }
}

void loop() {
  // Üres loop, hogy ne ismételje végtelenül
}
```