Laborator 1 - Sistemul de întreruperi

Coca Mihai
Ioana Dragoș

Tabelă de Conținut

Concepte introductive

Noțiuni de bază

Microprocessor *(single-chip CPU)* vs Microcontroller		
Componentă	**MPU**	**MCU**
Arhitectură	Von Neumann	Harvard
Memorie & I/O	Externe (c.extinse)	Interne (c. compacte)
CLK	Frecvențe mari ($\sim$ *GHz*)	Frecvențe mici ($\sim$ *MHz*)
Consum	Mare	Mic
Instrucțiuni	Operații externe	Operații interne
Programe	Complexe	Simpliste și specializate
SO	Suportă (>1GHz)	RTOS(deterministic)

Tabela Diferențele între microprocesor și microcontroller

Sistem embedded

- Este un ***sistem*** care utilizează unul sau mai multe microcalculatoare pentru a rula programe personalizate, dedicate și conectate la cel puțin o componentă hardware specializată, în scopul expunerii către utilizator a unui set dedicat de funcții
- PC-urile nu sunt proiectate în scopul rulării unui singur task dedicat unui set specializat de componente hardware
- **Exemple** - pilot automat masină, termostat AC, regulator semafor, controler sistem irigare, osciloscop, Mars Rover etc.

Microcontroller

- **Microcontroller** - componentă single-chip care conține pe lângă MPU, memorii de tip ROM, RAM, periferice (GPIO), timere și canale de comunicație serială, toate în același chip (“most compact design and the lowest hardware cost”)
- **Configurație** - dimensiune CPU (8, 16 maxim 32 biți), frecvență CPU (ordin 100 MHz)

Embedded Programming

- ***Initializare*** - startup code înainte de apelul main() & user program
- Un program creat pentru o platformă embedded va conține în general ***manipulări de biți***, atât pentru intrări cât și pentru ieșiri
- Un MCU va avea mulți regiștrii de ***configurare*** și de ***stare*** care sunt manipulați prin setarea, ștergerea, citirea unor biți individuali
- Fiecare ***registru*** poate fi găsit la o adresă fixă de memorie
- **Cunoștiințe necesare** - o bună înțelegere a echipamentelor de depanare, precum: multimetru, osciloscop, analizator logic etc.
- ‘all microcontroller programming is embedded programming’

Sistemul de întreruperi

Întrerupere

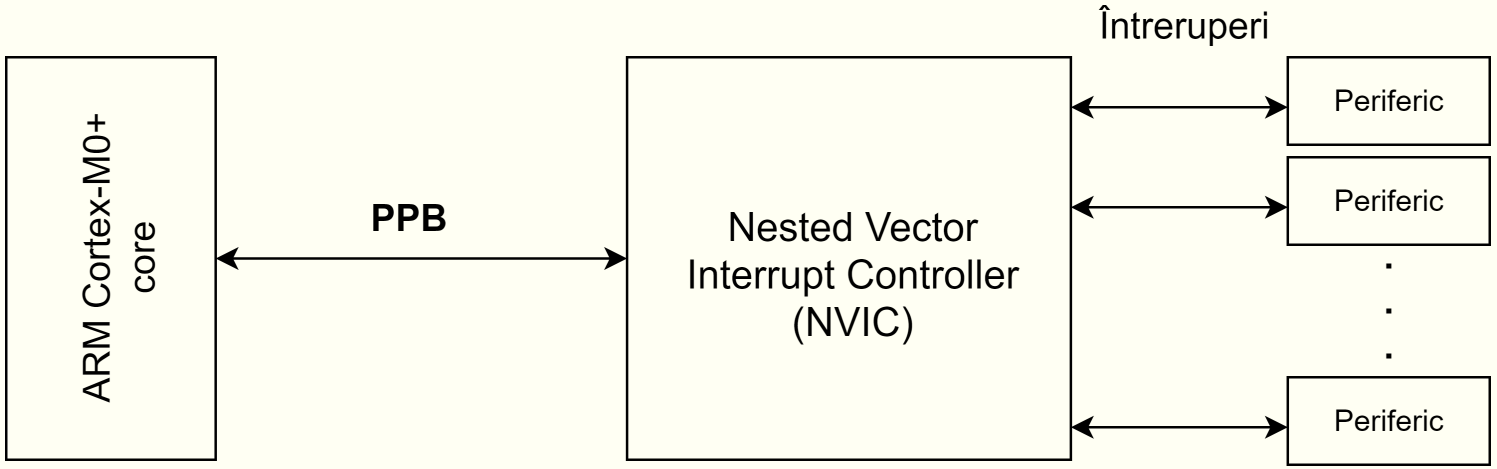
- ***Întrerupere*** - semnal transmis procesorului, emis fie de către un eveniment hardware/software, cu scopul execuției imediate (după terminarea instrucțiunii curente) a unei rutine de întreruperi
- ***ISR - Interrupt Service Routine*** reprezintă secvența de cod *invizibilă* (este salvat contextul codului întrerupt, iar după execuția rutinei este refăcut) executată ca răspuns la apariția unei anumite întreruperi
- Întreruperile mascabile pot fi activate/dezactivate

Tipuri de întreruperi

- ***Hardware*** - asincrone, nu sunt relaționate de codul curent executat de procesor
- ***Software*** - sincrone, sub formă de excepții, reprezintă rezultatul unor instrucțiuni specifice

NVIC

- ***Nested Vectored Interrupt Controller***



- Reprezintă o *tabelă* (vector) a adreselor rutinelor de tratare a întreruperilor (*ISR*) împreună cu numărul asociat și are rolul de a manageria și prioritiza întreruperile externe

NVIC

- Execuția codului curent este suspendată
- Contextul programului este salvat
- NVIC inițiază un apel către vectorul de adrese, localizează numărul întreruperii care a avut loc, extrage adresa rutinei de întreruperi
- Secvența de cod asociată rutinei este executată
- Contextul este readus și execuția programului este reluată

Sistemul de întreruperi

- Un MCU este proiectat în general pentru a răspunde la un număr de surse diferite de întrerupere (10 – 100 de surse). Fiecare sursă poate avea un cod specific scris de programator care este executat atunci când o întrerupere este desclanșată
- Programatorul decide care întreruperi vor fi active, precum și timpul de activare/dezactivare al acestora în execuția programului
- **startup_MKL25Z4.s** - conține tabela vectorilor de întreruperi, având 32 de handlere pentru întreruperi externe

Sistemul de întreruperi

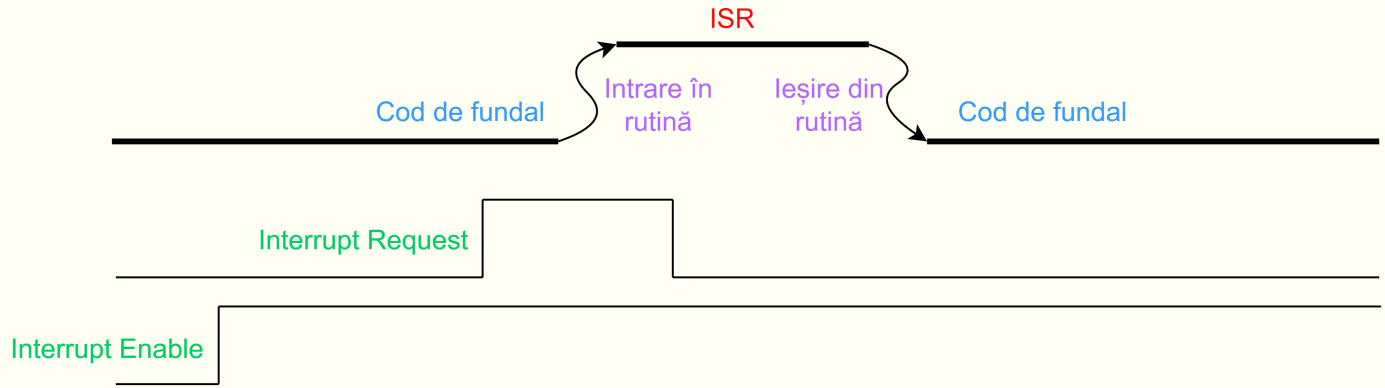
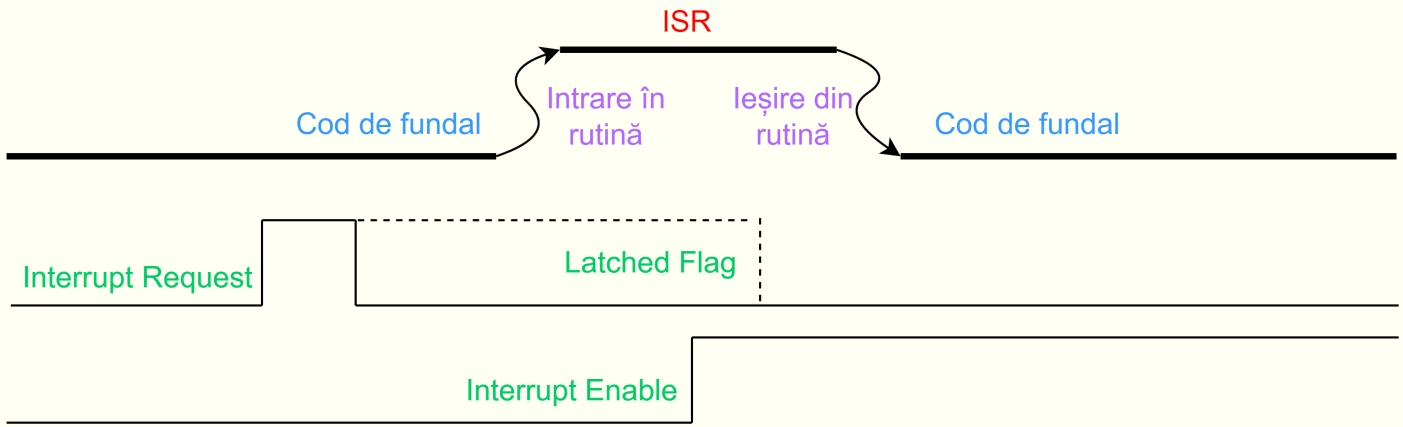


Figura Întreruperea codului executat la apariția unei întreruperi

- TVI va conține adresele de start a ISR-urilor asociate. Pentru întreruperile care nu vor fi active, intrările echivalente pot fi lăsate necompletate sau să pointeze către un ISR fictiv/default
- O funcție ISR trebuie să facă cât mai puțin posibil (menținerea unei latențe cât mai mici într-un sistem cu multiple surse de întrerupere active)

Sistemul de întreruperi

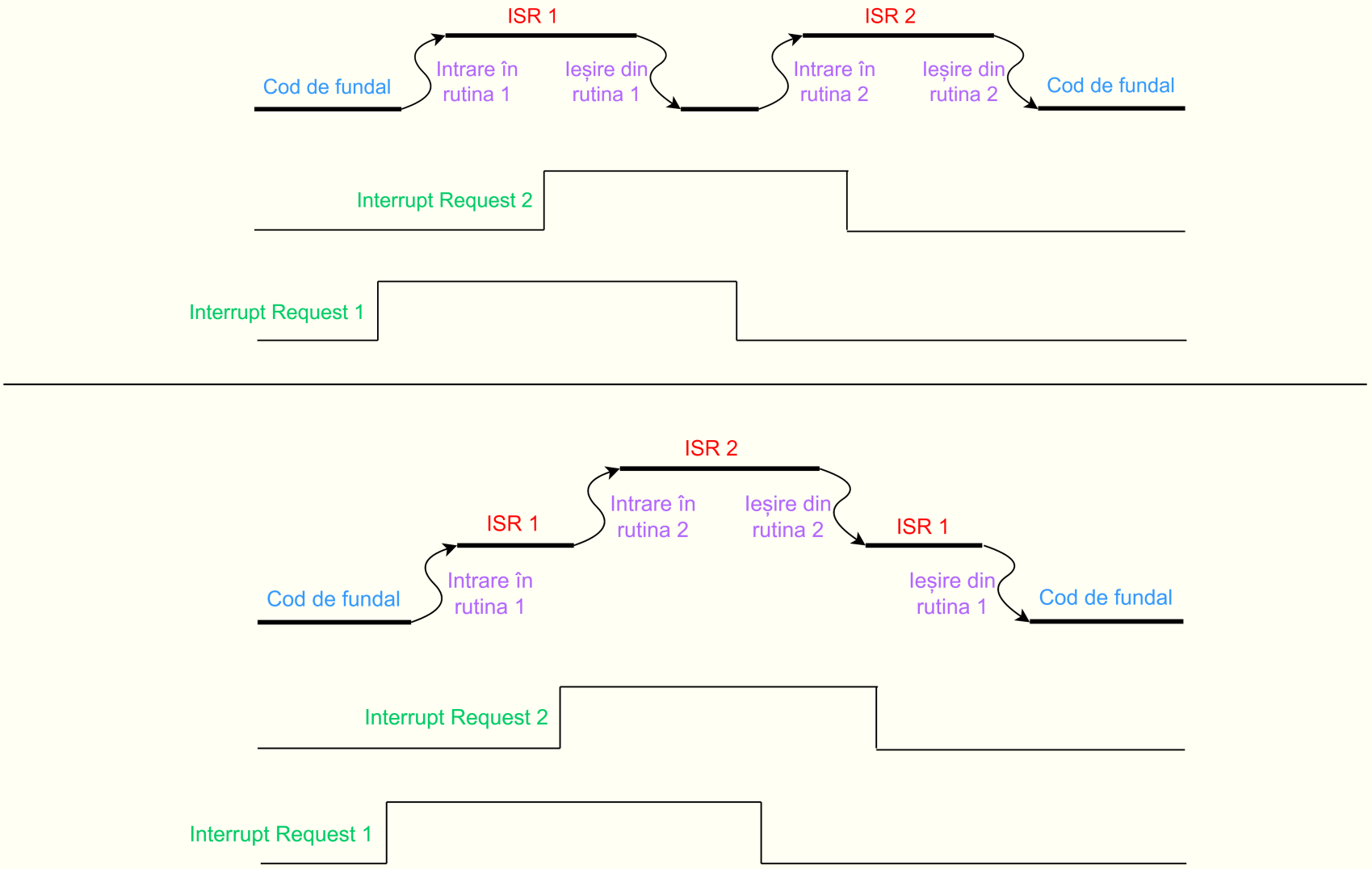
- În mod normal, o cerere de întrerupere este salvată (*latched*), ceea ce înseamnă că evenimentul de întrerupere setează un flag, a cărui stare persistă chiar dacă evenimentul este încheiat



Întreruperi multiple

- Într-un sistem cu multiple surse de întrerupere active, pot să apară situații de concurență asupra execuției ISR-ului asociat (***scenariu***: INT1 ISR se află în execuție și apare o cerere de întrerupere INT2)
  1. INT2 este blocată până când INT1 ISR se execută integral
  2. INT2 este mai prioritară decât INT1, iar INT1 ISR va fi întrerupt (se consideră faptul că INT2 va fi activată in interiorul INT1 din moment ce la intrarea într-un ISR, există o dezactivare globală a tuturor întreruperilor din sistemul respectiv)

Întreruperi multiple



Sistemul de întreruperi

- Marcarea secțiunilor critice într-un scenariu de suprapunere a accesului la o zonă de memorie partajată (acces neîntrerupt la date prin dezactivarea întreruperilor cu acces la zona respectivă)
- În funcție de compilator, dacă o variabilă este folosită atât în mainline code cât și într-o funcție ISR, variabila respectivă trebuie declarată ***volatile***. Altfel, componenta de optimizare a compilatorului va genera un output inconsistent

Sistemul de întreruperi

- ***Scenariu***: de obicei funcțiile ISR modifică variabile care sunt testate în mainline code. De exemplu, o întrerupere pe un port serial setează un flag global

```
bool gb_etx_found = false;

void rx_isr(void) {

  //...
  if(0x0D == rx_char){
    gb_etx_found = true;
  }
  //...
}

void main() {

  while(!gb_etx_found){
    // wait ...
  }
  INSTR1
}
```

Sistemul de întreruperi

- Compilatorul nu știe că variabila *gb_etx_found* ar putea fi modificată în ISR din moment ce nu apare niciun apel către ISR. Compilatorul observă că expresia ***!gb_ext_found*** va fi întotdeauna true, rezultând într-o buclă infinită, eliminând astfel din executabilul final INSTR1 (unreachable code)

Sistemul de întreruperi

- ***Edge-triggered*** - dacă apăsăm o tastă din tastatura telefonului pentru mai multe secunde, un singur caracter va fi inserat (o acțiune este efectuată doar dacă semnalul suferă o tranziție $\mathbf{0} \rightarrow \mathbf{1}, \mathbf{1} \rightarrow \mathbf{0}$)
- ***Level-triggered*** - dacă apăsăm o tastă din tastatura PC-ului pentru mai multe secunde, vom obține un string de caractere (o acțiune este efectuată atât timp cât un semnal este activ în funcție de polaritate, **0** sau **1**)