Coding Tomorrow Cup Döntős Feladat

Flash Driver

|  |
| --- |
| The copyright of this document and all supplements remain to thyssenkrupp AG. It is strictly prohibited to make copy of it and to distribute or make it accessible to a third party without a written authorization from thyssenkrupp AG. |

Tartalomjegyzék

[1. Bevezető 3](#_Toc513568868)

[2. Feladat kiírás 5](#_Toc513568869)

[2.1 Bemenő adatok 5](#_Toc513568870)

[2.2 Implementáció 5](#_Toc513568871)

[2.3 Fejlesztő és teszt környezet 6](#_Toc513568872)

[2.3.1 Teszt környezet elemei 6](#_Toc513568873)

[2.4 Megoldás benyújtása 6](#_Toc513568874)

[3. A Feladat pontozása 6](#_Toc513568875)

# Bevezető

Az autóiparban a beágyazott vezérlőegységek (ECU) beállításait és adatait nem felejtő részegységekben tároljuk, mely lehet EEPROM, vagy flash alapú nem felejtő memória. A flash alapú memóriák az ECU-ban alkalmazott processzorban közvetlenül integrálva is megtalálhatók. Ezen integrált flash memóriák előnye a költséghatékonyság. Az integrált flash memória a különböző processzorokban más és más címterületeken és különböző méretben érhető el (ezen kívül még az is igaz, hogy egy adott processzorban sem lehet feltétlenül egybefüggő területen a flash memória, tehát különböző méretű darabokban érhető el). Az autóiparban a költséghatékonyság elve azt is megkívánja, hogy a már megírt és letesztelt ECU szoftvert a lehető legkevesebb szoftverváltoztatással átvihető legyen egy másik processzor platformra.

A vezető autóipari szabvány az AUTOSAR, az ilyen integrált flash memóriák platform független elérésére kínál egy hatékony módszert (részben a felsőbb rétegbeli modulok komplexitásának csökkentésére, másrészt pedig ezen modulok könnyebb újrahasznosítására), mely egy egybefüggő virtuális címtérbe helyezi a felhasználni kívánt mennyiségű flash memóriát. Az említett módszer a *FlashDriver*-nek nevezett modul felsőbb rétegbeli funkcionalitását írja le.

Az alábbi ábrán látható példa a virtuális és fizikai címtér összerendelést szemlélteti. Az ábrán fel vannak tüntetve az egyes memória szekciókhoz (szeletekhez) tartozó kezdő címek mind a fizikai, mind pedig a virtuális címtérben, ill. az egyes szekciók méretei is.

Szekció 1 (4KByte)

Virtuális címtér

Fizikai címtér

Szekció 2 (4 KByte)

0x00000000

0x00001000

Szekció 3 (8Kbyte)

Szekció 4 (8Kbyte)

Szekció 5 (2KByte)

0x00002000

0x00004000

0x00008000

0x00008800

Szekció 1 (4KByte)

Szekció 2 (4 KByte)

0x80001000

Szekció 3 (8Kbyte)

Szekció 4 (8Kbyte)

Szekció 5 (2KByte)

0x80002000

0x80005000

0x8000A000

0x8000A800

Nem használt terület

Nem használt terület

0x80000000

0x80003000

0x80006000

Szekció 6 (2KByte)

Szekció 6 (2KByte)

0x00009000

0x8000B000

1 Virtuális és fizikai címtér összerendelés példa

# Feladat kiírás

Az alábbi feladatban egy képzeletbeli mikroprocesszor integrált flash memóriájának adott területeit szeretnénk adattárolásra használni oly módon, hogy az integrált flash memória területek (melyek a processzor fizikai címterületén helyezkednek el) egy 0-val kezdődő „virtuális” címterületen, egybefüggően legyenek írhatóak.

A megírandó algoritmus a bevezetőben említett AUTOSAR szabványban használt modul felsőbb rétegében található funkcionalitást valósítja meg. Mivel ezek az AUTOSAR modulok széleskörűen konfigurálhatók, a bennük megvalósított algoritmusokat is ennek megfelelően kell előkészíteni, tehát a *feladatban megvalósítandó algoritmusnak is kellően rugalmasnak kell lennie azokra az esetekre vonatkozóan is, amikor (i) megváltozik a szekciók száma, ill. (ii) a virtuális címtérben a memória szekciók nem feltétlenül a fizikai címtérben megadott sorrendben követik egymást*. A feladat egy későbbi kiegészítésében, majd találkozhattok ilyen feltételekkel, tehát az algoritmus mindenképpen figyelembe kell, hogy vegye ezeket, mert a kiegészítő feladatban az algoritmus már nem módosítható, csak a bemenő adatok, tehát a megvalósított algoritmus nem tartalmazhatja a bemenő adatokat.

## Bemenő adatok

A processzor gyártók a lehetőségeikhez mérten megpróbálják a saját beágyazott processzoraikban alkalmazott flash memóriák címterét egy kezdőcímtől kezdve lineárisan, hézagok nélkül feltölteni.

A helyzet az, hogy ez most sajnos nem sikerült valami jól, így kénytelenek leszünk az alábbi címkiosztáshoz konfigurálni az algoritmusunkat.

A képzeletbeli processzorban a flash memória szekciók a következőképen lettek elhelyezve. Az alábbi táblázat az egyes memória szekciók fizikai kezdőcímét és méretét adja meg.

|  |  |  |
| --- | --- | --- |
| **Szekció azonosítója** | **Szekció fizikai kezdőcíme** | **Szekció mérete (Bájt)** |
| 1 | 0x80000000 | 0x2000 |
| 2 | 0x80002000 | 0x4000 |
| 3 | 0x80008000 | 0x1000 |
| 4 | 0x80009000 | 0x2000 |
| 5 | 0x8000E000 | 0x8000 |

1 Memória szekciók azonosítója, fizikai kezdőcímei és méretei.

A virtuális címtérben a megadott szekciók (az azonosítójuk szerint) a következő sorrendben helyezkednek el: **1, 2, 3, 4, 5**.

A feladat megoldásához más paraméterre nincs szükség.

## Implementáció

A megírandó algoritmusnak egy előkészített függvényben **void FlashDriver\_Write(uint32\_t aVirtualAddress, uint8 aData)** kell helyet foglalnia, melynek paramétereiben a virtuális címtérben címezhetünk meg egy 1 bájtos memória cellát, melyet az algoritmus a flash memória fizikai címén keresztül ír, melyhez a **ModuleTest\_WritePhisicalFlashAddress(uint32\_t aPhysicalAddress, uint8\_t aData)** nevű függvényt kell majd az algoritmusnak meghívnia. Az előbbi függvény „ModuleTest\_” előtagja arra is utal, hogy ez a meghívott függvény az algoritmust is közvetlenül teszteli.

A **FlashDriver\_Write** függvényben az implementált kód csak a szükséges számításokat tartalmazhatja, tehát magát az algoritmust. A bemeneti adatokat paraméterként kell, hogy feldolgozza.

## Fejlesztő és teszt környezet

A fejlesztő és tesztelő környezet egy Eclipse projekt, mely C nyelven írt modulokat, header-eket és egy előre fordított függvény könyvtárat tartalmaz. A kapott Eclipse projektet nem szükséges módosítani, viszont a megfelelő fordítót (GCC) biztosítani kell. Saját fordító környezet is használható, de ebben az esetben nektek kell gondoskodni a projekt megfelelő fordításáról és a felmerülő problémák kiküszöböléséről, ami a feladat kidolgozásának idejéből vehet el értékes perceket. A kapott projekt módosítások nélkül is fordítható.

Az implementációt egy előre megírt modul teszt (libFlashDriverTest.a, vagy libFlashDriverTestLinux.a) segítségével tesztelhetjük le, mely ellenőrizni fogja az algoritmus által a **ModuleTest\_WritePhisicalFlashAddress** függvényhívás paramétereként megadott fizikai cím és adat helyességét. Ez az automatikus tesztelés már elő van készítve a kapott környezetben (main.c) és a **ModuleTest\_Start(),** *aType* = **MODULE\_TEST1** paraméterrel való függvény hívással indítható.

A modul teszt végeztével a konzol kimeneten megjelenik a teszt eredménye, mely a „**PASSED**”, vagy a „**FAILED**” üzenetet tartalmazhatja. „**PASSED**” esetén a feladat megoldása sikeres. „**FAILED**” esetén tovább kell tökéletesíteni az algoritmust.

## Teszt környezet elemei

* **FlashDriver.h:** Ebben a C header-ben található a megvalósítandó API (Application Programming Interface) függvény deklarációja.
* **ModuleTest.h**: Ebben a C header-ben találhatók a teszt környezet függvény deklarációi.
* **Types.h**: Ebben a C header-ben az alaptípus definíciókon felül a *boolean* típus, és a *NULL\_PTR* típus definíció is megtalálható és természetesen bárhol az implementációban fel is használható.
* **FlashDriver.c**: Ebben a C modulban található az a függvény váz, melybe az algoritmus implementációt el kell készíteni.
* **main.c**: Ebben a C modulban található a tesztet futtató függvény, melynek paraméterében csak a **MODULE\_TESTx** paramétert kell majd a későbbiekben módosítani.
* **libFlashDriverTest.a**: Ez a szoftver könyvtár tartalmazza az algoritmust tesztelő eljárásokat.

## Megoldás benyújtása

Amennyiben a feladat megoldásotok sikeres, a kiegészítő feladat elkezdése előtt küldjétek el az elkészült algoritmus kódját.

# A Feladat pontozása

* Működő algoritmus megvalósítása az alapfeladatban: 15 pont.
* Kiegészítő feladat megoldása az alapfeladatban megírt algoritmus módosítása nélkül: 25 pont.