

NAND flash eszközök programozása XELTEK programozóval.

Az ismertető elolvasása előtt fontos megismerni néhány olyan fontos kifejezést, rövidítést, pl “Block (Blok),” “ECC (Hiba javítási kód),” “Main area (Fő terület)” és “Spare area (Tartalék terület)” melyek a NAND flash adatlapokban is mindig szerepelnek.

Nand eszközök kulcsszavai:

“Block,” - Blokk

“ECC,” – Hiba javítási kód

“Main area,” – Fő terület

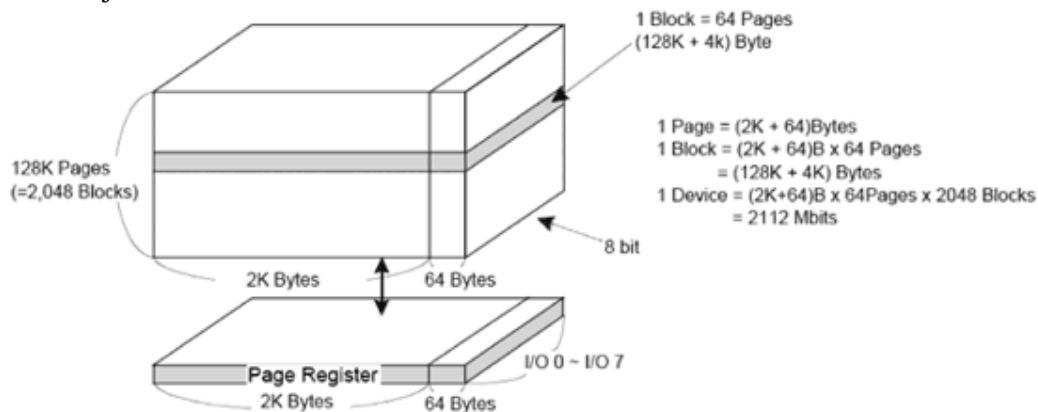
“Spare area,” – Tartalék terület

“RBA” – Lefoglalat blokk terület

“UBA” – Felhasználói blokk terület

1. Nand flash

A legtöbb NAND flash chip 512*8 (vagy 256*16) vagy 2048*8 (vagy 1024*16) bites „Fő területtel (Main area)” és 16*8 (vagy 8*16) vagy 64*8 (vagy 32*16) bites „Tartalék területtel (Spare area)” rendelkezik. Összesen 528*8 (264*16) vagy 2112*8 (vagy 1056*16) bit/oldal (page). Egy blokk, sok oldalból áll, de mivel a tartalék terület (Spare area) a hibás blokkok megjelölésére ill. az ECC (error-correcting code – Hiba javítási kód) értékeknek van fenntartva, csak a Fő terület érhető el a felhasználó számára. Egy 2-Gbit NAND eszköz, 2048 blokkba van rendezve, 64 oldal blokkonként (1. ábra). Minden oldal összesen 2112 bájtot tartalmaz, ami magában foglal 2048-bájt adat területet és 64-bájt tartalék területet.



1. ábra

A NAND flash eszközök előnye a NOR flash eszközökkel szemben, hogy nagy kapacitást biztosítanak, alacsonyabb áron. Továbbá, a NAND flash gyors írást (programozást) és törlést tesz lehetővé. A NOR flash előnye, hogy véletlenszerű elérés és bájt írási képességgel rendelkezik. A NOR flash eszközök véletlenszerű elérés képessége lehetővé tesz helyben történő futtatást is, mely számos alkalmazás esetén elengedhetetlen. A NAND flash eszközök hátránya a lassabb véletlenszerű elérés, viszont a NOR flash eszközök korláta a lassú írás és törlés végrehajtás.

A NAND alkalmasabb fájl szintű alkalmazások számára. Több processzor rendelkezik közvetlen NAND érdeklődéssel és képes bootolni közvetlenül a NAND flash eszközről (NOR nélkül).

A NAND hasonlít a merevlemezhez. Ez szektor-alapú vagy oldal-alapú és alkalmas arra, hogy

egymást követő adatokat tároljon, pl. képeket, hangokat, vagy PC adatokat. Továbbá, véletlenszerű elérés rendszer szinten végezhető el az adatok RAM-ba történő „árnyékolásával”, így egy kiegészítő RAM tároló is szükséges. Úgy mint egy merevlemez esetén is, a NAND eszközökön is vannak hibás blokkok, és az adatok épségének kezelése érdekében hiba javítási kód (ECC) szükséges.

Az új NAND flash is tartalmazhat érvénytelen blokkokat. További érvénytelen blokkok is kialakulhatnak a használat során. Az érvénytelen blokkok olyan blokkok amelyek egy vagy több hibás bitet tartalmaznak. A gyártó által megjelölt hibás blokkok törlése és programozása nem lehetséges.

A hibás blokkokat még a kiszállítás előtt azonosítják és a meghatározott területen megjelölik. A használat során további hibás blokkok alakulhatnak ki, melyeket a rendszernek kell majd azonosítania.

A bit átfordulásokat könnyebb a NAND flash eszközökben megtalálni. Ha bit átfordulás történik egy fontos fájlban, az a rendszer leállításához vezethet. Ezért az ECC/EDC algoritmusok ajánlatosak NAND flash programozáskor a megbízhatóság érdekében.

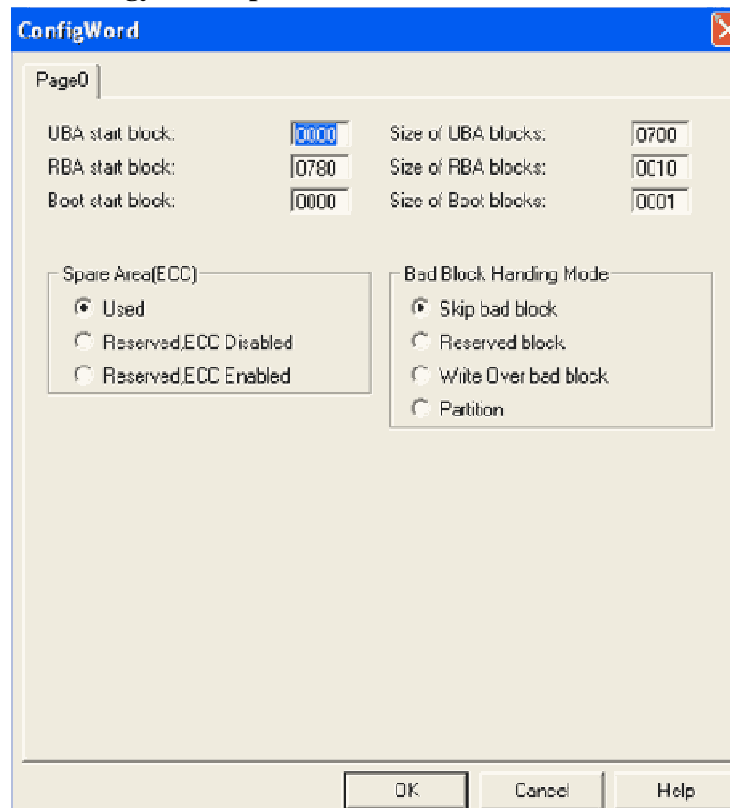
2. A hibás blokkok kezelése

Többféle hibás blokk kezelési módszer is létezik. A vásárló a saját igényeinek megfelelő módszert választhatja, vagy kérhet a Xeltektől egyedi igények szerint testreszabott algoritmust.

Beágyazott rendszer (embedded system) esetén a hibás blokkok kezeléséhez egy extra szoftver „réteg” szükséges. Ezért a NAND flash chipek programozásakor a hibás blokk kezeléshez pont ugyanazt kell tenni mint a felhasználói rendszerben is. Olyan egyszerű amilyennek hangzik, és több módon is kezelhető, miközben úgy tűnik nincs egy általánosan elfogadott univerzális megoldás sem.

Pl. az egyik leggyakoribb megoldás ha a hibás blokk egyszerűen kihagyásra kerül és az adat egy jó blokkra kerül. Ezt nevezik blokk kihagyás módszernek (Skip Block Method). Másik gyakori módszer (Samsung esetén), hogy a hibás blokkok, lefoglalt blokk területként (Reserved block area) kerülnek megjelölésre. Ezen a területen, egy blokk a hibás blokkok helyét tároló táblázat számára van fenntartva, a fennmaradó többi terület azokat az adatokat tartalmazza amik a hibás blokkra kerültek volna. Mindkét módszert tartalmazza a Xeltek NAND flash alap programozási algoritmus. 5-6 módszer érhető el a hibás blokk kezelésére. A két legáltalánosabb módszer alább kerül ismertetésre.

2.1. Hibás blokk kihagyás (Skip bad block)



2. ábra

A felhasználó a programozási módot az Eszköz Konfig (Dev.config) gombra kattintva választhatod ki.

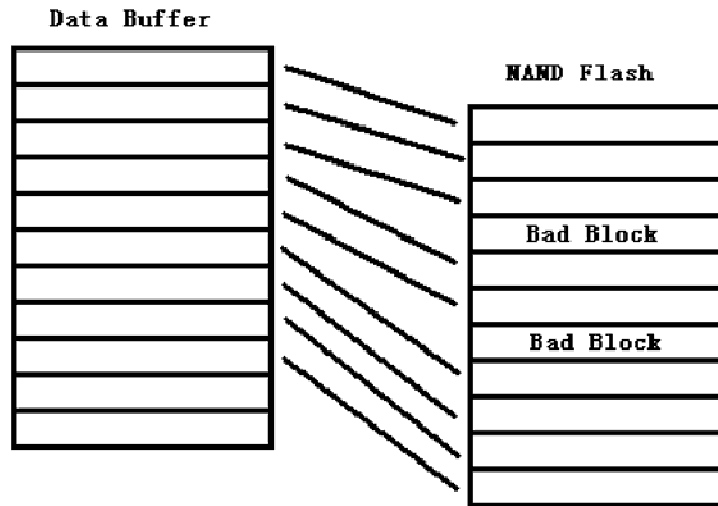
Megj.: Ha a felhasználói adat tartalmaz tartalék adatot (Spare data), más módszereknél később találsz információkat erről is.

A hibás blokk kihagyás módszere a legegyszerűbb, legáltalánosabb és legcélratoróbb módszer. Egyszerűen kihagyja a hibás blokkokat és az adatokat jó blokkokra helyezi. A hibás blokk jelölés kiolvasásával ellenőrzi az algoritmus a blokkokat. Ha a cél cím egy hibás blokk címére hivatkozik, akkor ez az oldal a következő jó blokkba kerül, miközben a hibás blokk kihagyásra kerül. Mivel a hibás blokk kihagyásra került, a tartalék területen található, az eredeti gyári programozási állapotú (nem FFh) adat jelzi a hibás blokk jelenlétét. Ezért a felhasználó rendszere képes, a boot ideje alatt, az összes blokk tartalék területének kiolvasásával egy új hibás blokk cím táblázat létrehozására.

A hibás blokk létezése miatt a felhasználói adat mérete nem haladhatja meg a jó blokkok méretét. A pontos értékeket add meg az “UBA start block (UBA kezdő blokk)” és “Size of UBA blocks (UBA blokk mérete)” mezőben.

UBA (User Block Area) : Felhasználói blokk terület

Néhány NAND eszközön nincs hibás blokk. Ezért a felhasználó az „UBA blokk méret” maximális értékét is megadhatja.



3. ábra

Ennek a módszernek a használata esetén, a felhasználónak ellenőriznie kell az UBA (Felhasználói Blokk Terület) kezdő blokkját, az UBA blokkok méretét. Lásd 2. ábra. A lefoglalt blokk terület (RBA) kezdő blokkját és az RBA blokkok méretét a lefoglalt blokk módszer esetén kell megadni. A Boot start blokkot és Boot blokk méretet akkor kell megadni, ha a boot területen is található hibás blokk. UBA kezdő blokk mutatja meg a chip programozás kezdő címét. Az alapértelmezett cím 0000. Az UBA blokkok mérete, a programozott blokkok száma.

2.2. Lefoglalt blokk (Reserved block)

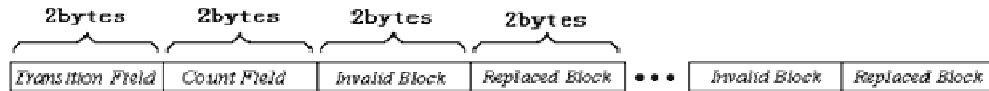
Ennél a módszernél egy hibás blokkok helyét tároló táblázat kerül létrehozásra. Hibás blokk érzékelésekor, a szoftver lefoglal egy jó blokkot annak cseréjére, és a lefoglalt területen létrehozza a szükséges leképező táblázatot.

Amikor chip kiolvasás közben hibás blokk kerül észlelésre, a szoftver kiolvassa a leképező táblát és ez alapján kiolvassa a kicserélt blokkot. Ennek az az előnye, hogy a tároló terület logikai épséggel rendelkezik. Ha egy hiba keletkezik az „A” Blokkban, egy külső tárolóból betöltve, megpróbálja az adatot átprogramozni egy másik blokkba („B Blokkba”). Ezután megakadályozza a rendszer számára a későbbi „A” blokk elérést, úgy hogy létrehoz egy hibás blokk táblát vagy felhasznál egy másik leképezési tervet. Ezt nevezik “Blokk helyettesítésnek” (Block Replacement).

A programozási algoritmus először meghatározza, hogy mely blokkok lesznek Felhasználói Blokk Területként (UBA) alkalmazva és, hogy mely blokkok lesznek Lefoglalt blokk Területként (RBA) használva. Az UBA kezdő címét és méretét az “UBA start block (UBA kezdő blokk)” ill. “Size of UBA blocks (UBA blokkok mérete)” mezőbe beírt paraméterek határozzák meg. Az RBA blokk kezdetét és méretét is a felhasználó határozza meg.

Az UBA és RBA kezdő címe között néhány blokk lefoglalásra kerül a hibás blokkok

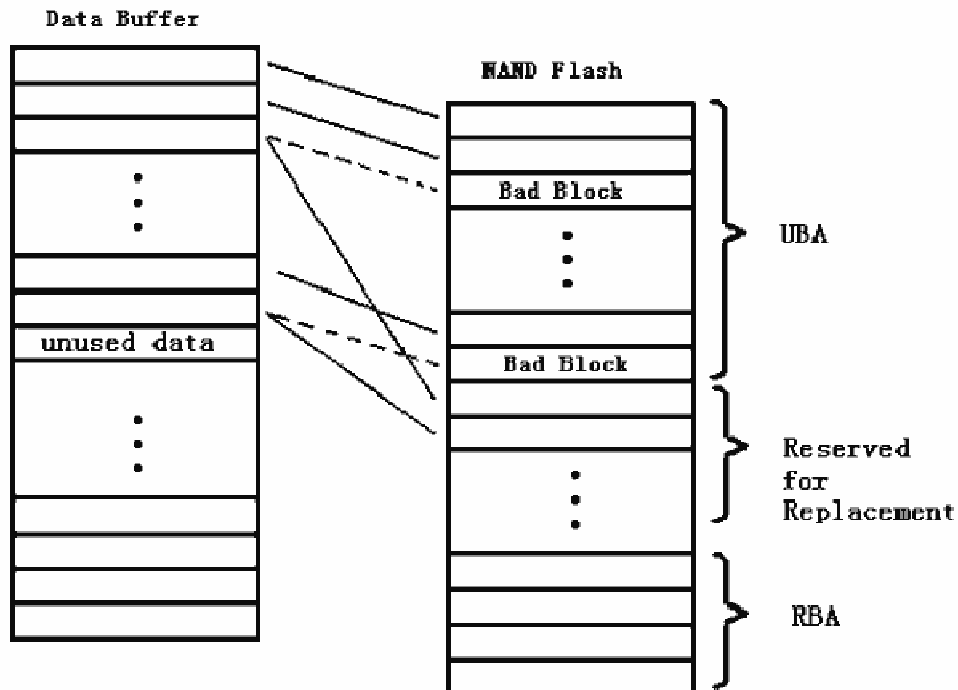
helyettesítésére. Az algoritmus kiolvassa az eszköz tartalék területét és létrehozza a „leképezési táblázatot” az RBA-ban. Csak az RBA első és második érvényes blokkja használható a táblázat számára. A táblázat tartalmazza azokat az információkat amelyek alapján az UBA-ban található hibás blokkokat a jókkal helyettesíti a lefoglalt területen. A táblázat adat mezője az alábbi.



A *Transition Field* (átmeneti mező) mindig FDFEh értékű. A *Count Field* (számlálás mező) táblázat oldalanként eggyel növekszik. A jelenlegi algoritmus számára csak két oldal használható, így az oldal leképező táblázatának első blokkja 0001h értéket, a második 0002h értéket fog tartalmazni.

Az *Érvénytelen blokk / Helyettesítő blokk* adat pár mutatja meg a hibás blokkok és a helyettesítő blokkok címét. Az oldal többi része tartalmazza ezeket az adat párokat. Mivel 512 bájt áll rendelkezésre, a maximális adat pár bejegyzés száma 127. Ez vélhetőleg elegendő mert a hibás blokkok száma általában az összes blokk 1%-a alatt van egy új, programozott eszköz esetén. Természetesen a használat során újabb blokkok válhatnak hibássá, így a rendszer (ezen módszer használatakor) azonosítja ezeket a blokkokat és frissíti a leképező táblázatot.

A leképező tábla második oldala az információk duplikálására van felhasználva, arra az esetre ha az első oldal véletlen megsérülne a használat során. Minden mező a “little endian” protokollt használja; az alacsonyabb bájt az első.



4. ábra

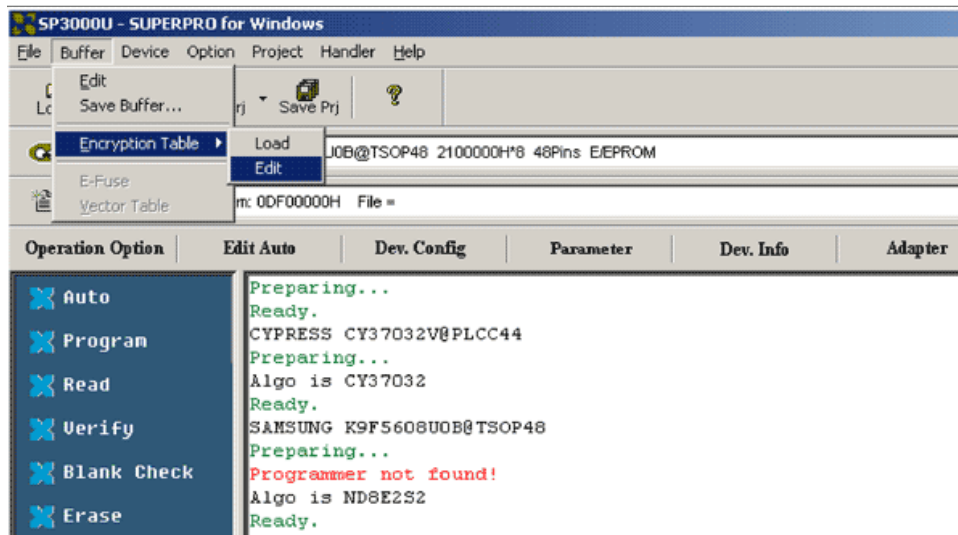
A 4. ábrán jelölt UBA a Felhasználói Blokk Terület (User Block Area), az RBA a Lefoglalat Blokk Terület (Reserved Block Area). A „Reserved for Replacement” (helyettesítési célból lefoglalt) jelölésű területen mely az RBA kezdőcíme és az UBA között helyezkedik el, helyettesítési célból lefoglalat blokkok vannak, melyek a hibás blokkok helyettesítésére szolgálnak.

2.3. Hibás blokk átírás (Write Over Bad Blocks)

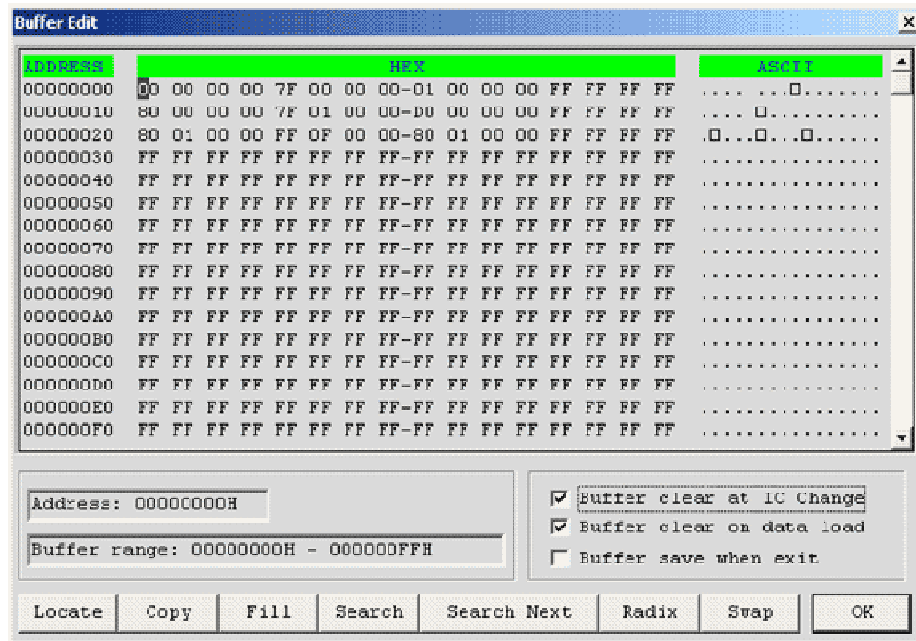
Ez a módszer, hibás blokk kezelő rendszer nélkül, a NAND teljes adat tartalmának tárolóba történő kiolvasására használható. Ezért a tároló tartalmazza a hibás blokkok adatait is. A felhasználó más külső cél programmal elemezheti és megkeresheti a hibás blokkok helyét. Ez a funkció nagyon hasznos ha a NAND egy harmadik fél által már programozásra került.

2.4. Partíció (Partition)

A partíció, a NAND Partíciós tábla alapú programozására használható. A partíciós tábla a programozás előtt kerül betöltésre és szerkesztésre. Lásd 5. és 6. ábra



5.ábra



6. ábra

A partíciós táblában minden 16 bájt jelent egy partíciót, amiben az 1-től a 4. bájtig jelenti a kezdő blokk címet, az 5-től a 8. bájtig a vég blokk címet, a 9-től a 12. bájtig a jelenleg használt blokk méretet, a 13-tól a 16. bájtig a megőrzött bájtokat. A cím keresés végetér amikor a kezdő cím értéke FF FF FF FF. A hibás blokk kihagyás módszer kerül alkalmazásra programozás közben. Pl. erre a chipre három partíciót kell programozni. Az első partíció kezdő címe 0000, a vég címe 0007F. Az aktuális blokk méret 0001.

3. Tartalék terület a NAND FLASH eszközön

Ha a felhasználói fájl tartalmaz tartalék adatot, ez a módszer a hatásos. Az algoritmus ellenőrzi a hibás blokk „flag” értékét az 1. és a 2. oldalon, és a nem-FF értékeket FF értékre cseréli.

A tartalék terület használata. Be kell állítani a 2. ábrán is látható Spare Area (ECC) (Tartalék terület (ECC)) opciót. Ha a programozandó fájl tartalmaz tartalék területre vonatkozó információt, válaszd a “Used (használva)” opciót. Ezt az opciót használni kell, ha az eredeti adatfájl információkat tartalmaz az eszközre vonatkozóan, ideértve a tartalék területet is. Ha nincs tartalék terület információ a programozandó fájlban, válaszd a “Reserved, ECC disabled (Lefoglalt, ECC letiltva)” opciót. Ha a “Reserved, ECC enabled (Lefoglalt, ECC engedve)” opció kerül kiválasztásra, az eredeti fájl nem fog tartalmazni tartalék terület adatot, de ha a felhasználó akarja, az eszköz tartalék terület programozásra kerül, ECC-vel vagy egyéb dinamikus adattal. A szoftver által létrehozott ECC algoritmust fogja a felhasználó használni. A XelteK programozóval használt alapértelmezett algoritmus az 512B ECC algoritmus. Az 1. táblázat mutatja meg a formátumot.

3.1. ECC (Hiba javítási kód) kiválasztása

Ha az "ECC" kiválasztásra kerül, az algoritmus az ECC adat három bájtját minden oldal 512 bájtjának megfelelően kiszámolja. Az ECC bájtok helye a 14., 15. és 16. bájt (másolással a 7., 8., és 9. bájt) a tartalék területen (pl.: 526, 527, és 528 bájt minden oldalon). Lásd 1. táblázat.

Table 1

1~5th	6th	7~9th			10~13th	14~16th		
Reserved	Bad Mark	ECC0	ECC1	ECC2	Reserved	ECC0	ECC1	ECC2



1 Byte

1. táblázat

Igény esetén a XELTEK programozó ki tudja számolni a tartalék területre mentendő adatot. Ebben az esetben adja meg igényeit a Xelteknak.

3.2. Hibás blokk megjelölés

Általában nagy oldal mód esetén, a hibás blokk jelölés minden blokk 2948. és 517. bájtjában van. Ezért a hibás blokk jelölés tartalma a programozandó fájlban FF. Ha nem ez a szabványos módszer került alkalmazásra a fájlban, a Xelteknak kérésre testreszabja az igényeknek megfelelően az algoritmust.

4. Boot Blokk ellenőrzés

A boot kódnak jó blokkokban kell elhelyezkednie, mert a legtöbb rendszer nem képes a hibás blokkok kezelésére a boot ideje alatt – mert a kód még nem töltődött be. A felhasználó meghatározhatja egy biztosan hibamentes terület kezdő blokkját és méretét, a "Boot start block (Boot kezdő blokk)" és "Size of boot blocks (Boot blokkok mérete)" értékek megadásával.

Ha a programozás során egy hibás blokk bukkan fel ebben a tartományban, a művelet végén egy hibaüzenetet küld a rendszer. A hiba lehetősége annál nagyobb, minél több „biztosan hibamentes területet” használunk. Ajánlatos a legkevesebb biztosan hibamentes területre beállítani a boot kódot.

Általában a boot rész nem tartalmazhat hibás blokkot. Ezért ellenőrizni kell a boot tartományt, hogy van-e benne hibás blokk. Ezután programozható a NAND chip. Ha a boot terület az első blokkban található, nem szükséges elvégezni a hibás blokk ellenőrzést mert a legtöbb NAND gyártó garantálja az első blokk hibamentességét. Ezért ha a kód elfér egy blokkban (és a biztosan hibamentes terület első blokkjának kezdő címét 0-ra állítod), a programozás során nem történik hiba.

A 2. ábrán látható a boot beállítása. A Boot start block (boot kezdő blokk) mezőben a boot terület kezdőcíme található. A Size of Boot blocks (boot blokkok mérete) mezőben a boot terület mérete található. A boot blokkok mérete akkorára legyen állítva mint az összes blokkok mérete, mert így lehetséges a Nand chip összes hibás blokkjának ellenőrzése.

5. Programozás

Programozáskor az ECC értékek mindig bekerülnek a tartalék területre. Az eszköz ellenőrzése és kiolvasása az ECC beállítástól függ. Az eszköz üresség ellenőrzése és törlése minden blokkon végrehajtásra kerül. A hibás blokk jelölések megóvása érdekében az üresség ellenőrzés (és szükség esetén a törlés) a programozás előtt végrehajtásra kerül.

Programozáskor ajánlatos az eszköz azonosítóját ellenőrizni és az automatikusan elvégzendő műveletek listájába behelyezni a "Programozást" és "Ellenőrzést". A NAND flash nagyon nagy mérete miatt időbe telik, eszköz kiválasztáskor, a fájl betöltése és művelet végrehajtása.

Új programozás mód, fájl rendszer, tartalék területre vonatkozó információk támogatása érdekében, kérjük add meg a részletes igényeket. A Xeltek megpróbál az igényeknek eleget tenni.

- Új hibás blokk feldolgozási mód.
- Nem szabványos hibás blokk jelölés.
- ECC algoritmus használat, kivéve ECC512B & ECC256W.
- Mentési szektor, blokk használat idő, programozandó fájl vagy egyéb információ a tartalék területre.
- FFS támogatás igény.

6. Gyakran Ismételt Kérdések

Kérdés: ECC hiba (ECC Error)

Válasz: ECC érték különbözik az eszköztől.

Kérdés: Nincs elegendő jó blokk (Good blocks not enough)

Válasz: A felhasználói adat nagyobb mint a jó blokkok mérete.

Kérdés: RBA sikertelen (RBA fails)

Válasz: RBA érték nem olvasható vagy ellenőrizhető.

Kérdés: Boot sikertelen (Boot fails)

Válasz: Nincs elegendő jó blokk.

Kérdés: Javítható ECC hiba (ECC_CORRECTABLE_ERROR)

Válasz: Keresd meg az ECC_CORRECTABLE_ERROR bitet és javítsd.

Kérdés: Az eszköz alapértelmezett beállítása szerint, az UBA blokk mérete 1F00, az RBA blokk mérete 0010 értékre lett beállítva. Ennél a beállításnál az utolsó 256 blokk nem látható. Nem használok semmilyen „lefoglalat blokk terület” típust sem, így az UBA Blokk méretet 0X2000 értékre, minden más beállítást 0-ra állítottam. A hibás blokk kihagyása (Skip Bad Blocks) került kiválasztásra. A programozó (superpro 3000U) egy "Good blocks not enough (Nincs elegendő jó blokk)" hibaüzenetet küld.

Válasz: Ha ki akarod olvasni a chip minden területét, először ellenőrizd a hibás

blokkokat. Ezután azokat vond ki az összes blokkból. Ezek az aktuális kiolvasandó blokkok.

Kérdés: Van lehetőség a hibás blokkok számának ellenőrzésére?

Válasz: A konfigurációs ablakban a Boot Start Block (Boot kezdő blokk) mezőbe írd 0000-t, a Size of Boot blocks (Boot blokkok mérete) mezőbe 2000-t. Kattints a Boot Block Check (boot blokk ellenőrzés) menüre és a hibás blokkok száma kiírásra kerül.

Kérdés: Konfig hiba, a művelet folytatása nem megengedett (Config Error, not allow continuing operation)

Válasz: Ez azért történik mert a konfig ablakban megadott értékek nem megfelelően lettek beállítva. A helyes beállítás az alábbi:

UBA kezdő blokk < Összes blokk, UBA blokkok mérete <= (Minden blokk - UBA kezdő blokk)
Boot kezdő blokk < Összes blokk, Boot blokkok mérete <= (Összes blokk - UBA kezdő blokk)

Lefoglalt mód esetén:

UBA kezdő blokk < RBA kezdő blokk, UBA blokkok mérete < (RBA kezdő blokk - UBA kezdő blokk)
Összes blokk > RBA kezdő blokk > (UBA kezdő blokk + UBA blokkok mérete)
RBA blokkok mérete < (Összes blokk - RBA kezdő blokk)

Kérdés: Programozáskor az alábbi üzenet jelenik meg “Good blocks not enough (Nincs elegendő jó blokk)”

Válasz: Ez a probléma akkor jelentkezik ha a programozandó fájl nagyobb mint a chip összes jó blokkjának mérete.

A konfigurációs ablakban a chip aktuális méretének megfelelő értéket kell beállítani. Pl. Kiolvasol minden blokkot=0x2000, de az eszköz rendelkezik hibás blokkokkal, így a jó blokkok száma kevesebb mint 0x2000.

UBA blokkok maximális mérete = Összes blokk – hibás blokkok.

Ha pl. 16 hibás blokk található a chipen, az aktuális jó blokk mérete egyenlő kell legyen: (Összes blokk – 0x10).

Kérdés: Használható a XelteK programozó olyan NAND eszköz kiolvasására amit más rendszeren programoztak?

Válasz: Ha egy, már más által programozott NAND flasht kell kiolvasni, azt a jó ill. hibás blokkok figyelembevétele nélkül megtehetjük, az Eszköz konfig ablakban található “Write over bad blocks (Hibás blokkok átírása)” opció kiválasztásával. Ha ez a módszer kerül használatra, az eszköz minden adata a tárolóba kerül kiolvasáskor anélkül, hogy figyelembe venné a hibás blokk kezelési módokat. Ezért a tároló tartalmazni fogja a hibás blokkok tartalmát is. Speciális programmal a hibás blokkok azonosíthatók az adatban.

Kérdés: Mi a feladata a NAND flash vezérlőnek?

Válasz: A NAND flash eszközök véletlenszerű adat elérése nem lehetséges, az adatok csak sorosan érhetők el. Ha egy memória oldal megnyitásra kerül, az adat könnyen és

gyorsan kiolvasható. A parancsokat sorosan kell elküldeni a NAND flash csatolóhoz (ONFI). Közvetlen NAND kapcsolatot létesíteni nehéz. A NAND vezérlők teszik könnyebbé a NAND flash elérését és egy letisztult, használható kezelőfelületet nyújt a memóriához. A vezérlők támogatnak opcionális hiba javítási kódot (ECC), ami egy-bites hiba javítást hajt végre.

Kérdés: Mi a különbség az MLC és SLC NAND között?

Válasz: Az MLC (Multi Level cell - Több szintű cellás) NAND flash két vagy több információ bitet tárol egy cellában. Az SLC (Single Level cell – Egyszintű cellás) NAND egy bitet tárol cellánként. Az MLC NAND nagyobb tároló kapacitással rendelkezik, de lassabb és kevésbé tartós. SLC = 100,000 törlés/programozás ciklus. MLC=10,000 törlés/programozás ciklus. Az MLC 4 bit ECC-t igényel, míg ellenben az SLC 1 bit ECC-t igényel.

Kérdés: Mi a különbség a nagy blokkú NAND flash és a kis blokkú NAND flash között?

Válasz:

Nagy blokk=> (2048 adat bájt +64 bájt tartalék adat) oldalanként.

Kis blokk=> (612 adat bájt +16 bájt tartalék adat) oldalanként.

A parancs sorok különböznek nagy blokkú és kis blokkú NAND-ok esetén.

Kérdés: Mi az a tartalék terület a NAND flashban?

Válasz: A nagy vagy kis blokkú NAND flash esetén 64 ill. 16 bájt tartalék terület található oldalanként. A tartalék terület tárolja az ECC-t és a többi hibás blokk kezelési információt (hibás blokk jelölés).

Kérdés: Mit értünk hibás blokk alatt?

Válasz: Nem minden memória cella működik tökéletesen egy NAND eszközben. Néhány blokk (1 blokk = 64 oldal) hibás blokként kerül meghatározásra. Tesztelés során, a hibás blokkok a NAND flash első két oldalának tartalék területén kerülnek megjelölésre. A felhasználónak ki kell olvasni a hibás blokk jelöléseket, és el kell kerülni azok használatát. A NAND flash használata során további hibás blokkok is keletkezhetnek. A felhasználónak kezelnie kell az új hibás blokkokat is.

Kérdés: Mi az a használat rétegzés (wear leveling)?

Válasz: A NAND flash korlátozott számban törölhető és programozható. Ha egy oldal gyakran kerül elérésre, az oldal memória cellája megsérülhet, így az egész NAND flash válik használhatatlanná. A használat rétegzéssel kiegyensúlyozható a NAND flash kihasználás és szétosztja a memória használatot a NAND flash minden oldala között. Ez segít a NAND flash élettartamát meghosszabbítani. Ez magába foglalja a NAND eszköz logikai címeinek különböző fizikai címekre történő kiosztását is.

Kérdés: Hogy tudok lemásolni egy már (más rendszeren) programozott NAND chipet?

Válasz: A NAND flash lemásolása (duplikálása) bonyolult ha a cél eszköz tartalmaz hibás blokkokat. A NAND flash másolás különbözik a szokványos EEPROM vagy

flash másolástól, mert a NAND flash chippek hibás blokkokat tartalmaznak.

Pl. ha a **K9F1G08U0A** NAND flash egy más rendszeren már programozásra került, kiolvashatjuk anélkül, hogy foglalkoznánk a hibás ill. jó blokkokkal. A Konfigurációs ablakban válaszd a **“Write over bad blocks/hard copy (Hibás blokk átírás/ Teljes másolás)”** opciót. Ha ez a módszer kerül használatra, az eszköz minden adata a tárolóba kerül kiolvasáskor anélkül, hogy figyelembe venné a hibás blokk kezelési módokat. Ezért a tároló tartalmazni fogja a hibás blokkok tartalmát is. Nagy gyártási mennyiség esetén ez nem ajánlott mert a cél chippek tartalmazni fogják a hibás blokkokat is.

Ha más eszközzel már programozásra került a NAND flash, a chip másolás a teljes másolással elvégezhető, ami két lépésre osztható – eredeti eszköz kiolvasása és cél eszköz programozása.

1) Eredeti eszköz kiolvasása: Ha az eszköz konfigurációs ablakban a „hard copy (teljes másolás)” opció kerül kiválasztásra, figyelmen kívül hagyjuk a jó és hibás blokkokat. Ha ez a módszer kerül használatra, az eszköz minden adata a tárolóba kerül kiolvasáskor anélkül, hogy figyelembe venné a hibás blokk kezelési módokat. Ezért a tároló tartalmazni fogja a hibás blokkok tartalmát is.

2) Cél eszköz programozása: A sikertelen programozást a cél eszköz hibás blokkjai okozhatják. Ezért a cél eszköznek teljesen hibamentesnek kell lennie.

A hibás blokkok száma leellenőrizhető, ha a konfigurációs ablakban engedélyezésre kerül a boot check assert (boot ellenőrzés megkövetelése). Ehhez a Boot Start Block mezőbe írd 0000-t, a Size of Boot blocks (boot blokkok mérete) mezőbe az összes blokkot (800). Kattints a szoftver panelon a „Boot Block Check”-re (Boot blokk ellenőrzés), és a hibás blokkok száma kiírásra kerül. Teljes másolás (Hard copy) módszere nagy gyártási mennyiség esetén nem ajánlott. Ebben az esetben küldd el a Xelteknek az adatfájlt, a kiegészítő információkkal, a használt tartalék területre és hibás blokk kezelésre vonatkozó információkkal együtt.

Kérdés: Egy 2GB-nál nagyobb NAND flasht kell programoznom. Ezt meg tudom tenni egy Xeltek SuperPro5000 programozóval?

Válasz: A 2GB-nál nagyobb NAND flash eszközök programozásához az SP6000 programozó ajánlott. Az SP6000 programozó 256Gb-ig méretű fájlokat is támogat. SP5000 esetén az eszköz több részletben történő programozásával, eltolás használatával végezhető el.