

Flash-alapú betöltőprogram „testre szabása”

A flash-alapú mikrovezérlők gyári betöltőprogramja egyéni igényekhez szabható

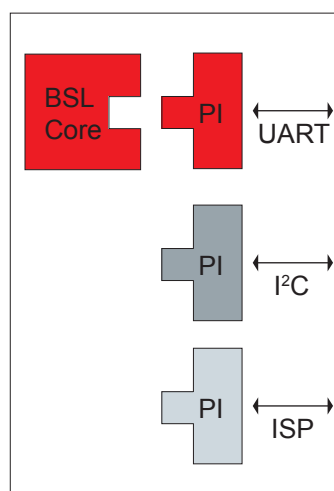
A bootstrap-loader fogalma a mikroprocesszor előtti kor „öreg harcosai” számára is ismerős lehet: ezzel töltöttük be a teljesen üres memóriájú számítógépekbe a futtatni kívánt programot. Ez a megoldás most a flash-memóriás mikrovezérlőknél „köszön vissza”: a Texas Instruments MSP430 MCU-család betöltőprogramja a felhasználói igényekhez igazítható.

A Texas Instruments (TI) MSP430F5xx mikrovezérlő családjá számos izgalmas újdonságot kínál. Ezek egyike, hogy az eszközök olyan beépített bootstrap-loader¹ (BSL) programmal rendelkeznek, amely segít betölteni a futtatni kívánt programot az eszköz flash-memóriájába. Az új 5xx BSL annyiban hasonlít a korábbi típusok ROM-ban tárolt, fix BSL-jéhez, hogy ez is lehetővé teszi, hogy egy külső hosztszámítógép egy sorozat végrehajtható utasítást küldhessen az eszköz belső memóriájába. Ezekkel az utasításokkal aztán különféle műveleteket végeztethetünk: átírhatjuk vagy törölhetjük a memóriát, programvégrehajrást kezdeményezhetünk, vagy éppen kiszámíthatjuk a memóriatartalom ellenőrző összegét (CRC-jét).

A TI jelenlegi BSL-programjai hasonlóak a korábbi modellek ROM-ban tárolt BSL-jeihez, ám az a tény, hogy a BSL-t jelenleg a flash-memóriában helyezik el, új, izgalmas lehetőséget ad a felhasználóknak arra, hogy maguk készíthessék el saját „testre szabott” BSL-jüket, amely személyes igényeiket jobban kiszolgálja. Ez megtehető úgy is, hogy felhasználják a gyári BSL részleteit, de a BSL akár teljesen újra is írható, amellyel a beágyazott mikrovezérlők „terepi” újraprogramozását valóban saját igényeihez igazíthatja a fejlesztő.

A testre szabási eljárás nem korlátozódik egyszerű programvégrehajrást (például kommunikációs interfész cseréjére vagy utasítások átvitelére). Ezen túlmenően szó lehet arról is, hogy a felhasználó a BSL egész életciklusát testre szabhatja, beleértve a BSL indítási kritériumait is. Minden rendszerindításkor egy BSL-funkcióhívás történik, amelynek az a feladata, hogy meghatározza, vajon a legközelebbi lépésben a BSL vagy a felhasználói programkód indítása történjék-e meg. A TI által programozott, UART-alapú BSL-nél ez egyszerűen annyit jelent, hogy az eszköz viselkedését az határozza meg, hogy a „BSL invoke” (BSL indítás) csatlakozóponton egy adott jelszekvencia jelent-e meg. Ez a funkció viszont könnyen testre szabható. Például az USB-alapú BSL-nél ez úgy változott meg, hogy a BSL egy gomb tartós megnyomásával akár oly módon is elindítható, hogy az üres memóriájú MCU az USB-csatlakozó felől kap tápellátást. Ez a rugalmasság lehetővé teszi, hogy az üres MSP430 eszközöket a gyártás során a létező USB-interfészükön keresztül beprogramozzuk. Azonban sok más kritériumnak is eleget tehetünk. Például az alkalmazás „bombabiztosságát” jelentősen javíthatjuk azáltal, ha a BSL-funkciókat kibővítjük egy CRC-ellenőrzéssel, mely a teljes alkalmazási flash-memóriára kiterjed. Ha ez a CRC-számítás nem ad egy előre meghatározott, megfelelő értéket, jelzés lehet arra nézve, hogy ismét a BSL-t kell elindítani a felhasználói program betöltésének megismétlésére.

A TI által szállított gyári BSL-t azzal a szándékkal készítették, hogy azt a felhasználó tetszés szerint alkalmazza, átalakítsa, azaz saját igényeihez igazítsa. Ennek megfelelően moduláris felépítésűnek írták meg, amelyben elkülönül a belső BSL „motor” és a kommunikációt irányító programrész. A BSL-motornak van egy háromfunkciós interfésze azokhoz a szoftverekhez, amelyek a kommunikációt fizikai szinten kezelik. Ezen a módon a BSL-motor egyszerűen inicializálja az interfészt, megmondja neki, hogy készüljön fel



egy adatsomag vételére, majd elküldi azt. Ezután lehetővé válik, hogy egyszerűen írassunk „testre szabott” BSL-t a kommunikációs programkód lecserélésével. Ha a szabványos, TI által szállított BSL valamilyen okból nem felel meg a célunknak, a kódot felülírhatjuk úgy, hogy az akár az I²C-, az SPI-, az RF- vagy az USB-interfészfelületen, vagy bármi más elképzelt interfésszel vagy protokollal kommunikáljon. Még ha csak olyan triviális feladatunk van is, mint egy portkivezetés felcserélése (például élve az 5xx MCU-k portkiosztási lehetőségével, vagy egy időzítőkivezetés „áthelyezése”), ezeket is könnyedén megoldhatjuk egy helyszíni rendszerfrissítéssel annak érdekében, hogy az eszköz jobban illeszkedjék a frissített rendszer követelményeihez.

„Belülről” nézve a BSL-motort úgy írták meg, hogy az lehetővé teszi a kód részleteinek „újrahasznosítását” vagy módosítását. A „motoron” belül két réteget különböztetnek meg: az első a parancsok dekódolásával foglalkozik annak eldöntése érdekében, hogy milyen teendőket kell végrehajtani. A második réteg foglalkozik közvetlenül az MSP430 memóriájába való fizikai írással és olvasással. Ez az alacsonyabb réteg afféle „forgalomirányító” az MSP430 memóriája felett: olyan egyszerű funkcióhívásokkal kezelhető, mint a writeMemory vagy a readMemory stb. Ez a forgalomirányító réteg foglalja magában azokat a biztonsági funkciókat is, amelyekkel az eszközbe írt program mint szellemi tulajdon megvédhető az illetékelten kiolvasástól, törléstől vagy módosítástól. Ez az a réteg például, amely fogadja az eszközt védő jelszót, és

¹ A bootstrap-loader elnevezésnek jó magyar megfelelőjét nem ismerem. Szó szerinti fordítása mulatságos lenne. A „csizmahúzó” kifejezés olyan egyszerű, majdhogynem primitív szoftver-segédesszközre utal, amely semmi más nem tud, csak „behúzni”, a memóriába befogadni a valóban érdemi funkciót végrehajtó programot. Az idősebbek emlékezhetnek, hogy egyes „ős számítógépeknél” (PDP-8, TPA-i stb.) ezt a bootstrap-loadert a „mérnöki pult” kapcsolóival „bitszinten”(!) kellett „bezongorázní” – a sokadik gépindítás után már fejből tudtuk a szekvenciát ©. A jó magyar kifejezés hiánya miatt azonban nem érdemes „könnyeket ejteni” – túlhaladta az idő: ma már szinte köznyelvi fordulat, hogy „lassan bútol a számítógép” – de a reggeli ébredés utáni kábulatról is hallottam már így beszélni: „kell egy kávé, különben nehezen bútolok”. – A szerk. megj.

törli az eszköz tartalmát, ha hibás jelszóval próbálkozik a jogosulatlan felhasználó. Ha viszont a jelszó helyes, ez a réteg nyitja meg az eszközt, és lehetővé teszi, hogy a továbbiakban memóriaolvasás vagy írás történjék. Ez a réteg meglehetősen „szimplán” van megírva, ezért valószínűleg bármilyen egyedi, testre szabott BSL-hez felhasználható. Az is az előnye, hogy ebbe az alsó rétegbe beépített biztonsági funkciók beépülnek bármilyen felhasználói igény szerint átírt, egyedi BSL szolgáltatásaiba, remélhetőleg kizárva az olyan közönséges biztonsági rések kialakulását, mint a hibás, biztonsági szempontból kiszolgáltatott BSL-állapotok fellépése, vagy a jelszó próbálgatásos feltörése. E réteg felhasználói igény szerinti átalakítása valószínűleg csak azoknak az esetenként túl szigorú „büntetéseknek” a kiküszöbölésére irányul, mint a hibás jelszó utáni teljes törlés vagy a teljes RAM felülírása a BSL indításakor.

A BSL-motor felső rétege egyedül csak a vett utasítások értelmezésével foglalkozik, és azoknak megfelelő funkcióhívásokat generál az alatta levő memóriairó/-olvasó szint számára. A szabványos TI BSL-ben ez az alsó réteg olyan parancsokat fogad, mint a memóriairás vagy -olvasás, CRC-számítás és a teljes memória törlése. Ellenben nincs akadálya annak, hogy további alkalmazáspecifikus parancsokat is beépíthessünk. Például olyan parancsot is definiálhatunk, amely titkosított bájtsorozat vételeit is lehetővé teszi. Ez a parancs jelezheti, hogy a vett adatsomag titkosított tartalmú, ezért mielőtt az eszköz memóriájába íránk, dekódolni kell a BSL-be épített, belső titkosítókulcs felhasználásával. Az ilyen utasítás például olyan tulajdonsággal ruházhatja fel a rendszert, hogy az eszközben tárolt szellemi érték akkor is

biztonságban marad, ha a weben keresztül, nyilvános hálózaton át történik a szoftverfrissítés.

Végül pedig, ha a BSL-re a végfelhasználói alkalmazásnak nincs tovább szüksége, törölhető is. A BSL törlése két különböző előnnyel jár. Először: megnyugtató lehet a tudat, hogy a BSL nélkül maradt eszköz 100%-osan védett bármilyen BSL-alapú támadással szemben. Másodszor, a BSL eltávolításával 2 kb-jattal több hely jut a felhasználói programkód számára. Ez a 2k memória „ingyen van”. Ráadásul ez a memóriahely olyan nem-BSL felhasználói kód céljára is használható, amely kihasználja ennek a BSL-memória céljára készült 2k méretű tárhelynek a speciális tulajdonságait, nevezetesen a törlés, kiolvasás és felülírás elleni fokozott védelemet. Ebben a 2k-ban tárolható a program biztonsági szempontból legkényesebb „magja”, vagy olyan firmware-elemei, amelyeket célszerű „módosíthatatlan” tárolóhelyen, magától az alkalmazástól elkülönítve tárolni. Ez lehet egy egyszerű funkciókönyvtár vagy akár egy jól kidolgozott alkalmazás is. A firmware két része közötti kapcsolattartás a BSL „Z-területén” keresztül történhet, amely átjáróként használható arra, hogy csak ellenőrzött körülmények között történhessen vezérlésátadás a biztonságos BSL-területre helyezett kód és a külső felhasználói program között.

Az olvasó további információkat és példákat találhat a „Creating a Custom Flash-Based BSL” (SLAA450) című alkalmazástechnikai dokumentumban a <http://www.ti.com/msp430/> webhelyen.

www.ti.com/hu

<http://www.ti.com/ww/hu/cikkek-szakirodalom.html>

DR MagLev – új generációs ventilátorok a Sunon cégtől

Új technológiai megoldások az élettartam növelése érdekében

A műszerventilátorok élettartamát jelentősen befolyásolja a porral szembeni ellenálló-képesség növelése és az olajszivárgás megelőzése, amit a Sunon cég új technológiák bevezetésével oldott meg.

1999-ben a Sunon cég elsőként mutatott be MagLev-motorral szerelt ventilátort, amely terméknek világszintű értékesítése azóta meghaladta a 600 millió darabot. A Sunon cég DR MagLev (Dust-Resistance MagLev)-motorral szerelt ventilátora az új műszaki megoldásokkal bővített MagLev-technológiát hasznosítja, amely növeli a porral szembeni ellenálló-képességet és megelőzi az olajszivárgást.

2009-ben – 8 évnyi tesztelés és vizsgálatok követően – a Sunon bemutatta a továbbfejlesztett DR MagLev-motorral szerelt ventilátort. Ezt a verziót az olajkifolyás elleni magasabb szintű tömítettség, a jobb porállóság, a nagyobb üzembiztonság és a hosszabb élettartam jellemzi.

