VCPU R1

VCPU programozási dokumentáció

V0.996

2022.04.07.+

1 Bevezetés

1.1 Célok

A *Flexible SD drive firmware*[1] (röviden *FlexSD fw*) VCPU[2] bővítés célja, hogy a klasszikus Commodore lemezes meghajtók mintájára az SD2IEC[3] meghajtót is alkalmassá tegye a beletöltött program futtatására. Az SD2IEC egy meghajtócsalád, ami több fajta hardveres felépítéssel rendelkezhet. A VCPU segítségével futtatott program egységes (virtuális) hardvert használ, így nincs szükség különálló támogatásra. Viszont a bővítésnek **nem célja egy, már meglevő, régi meghajtóval való kompatibilitás**! A VCPU nem csak a szokásos lemezképekkel tud dolgozni, a meghajtó összes funkcióját használhatja, így a hagyományos lemezeknél lényegesen nagyobb kapacitású háttértár válik elérhetővé a számítógép számára. Ezen felül implementálható a soros buszon bármilyen, a programozó igényeinek megfelelő rendszerű adatátvitel.

1.2 Eltérések az eredeti Commodore háttértárak programozásától

A meghajtó VCPU-s programozása az eredeti Commodore meghajtók direkt hardver programozásától némileg eltérő logikájú, itt a tárolást végző eszköz közvetlen kezelésére csak korlátozottan van lehetőség. A használható parancsok ugyanazok, amiket a számítógép kiadhat a soros buszon keresztül, de a VCPU-s programfuttatás alatt az ezeket a parancsokat kiadó program „beköltözik” a meghajtó memóriájába, ahova a végrehajtott feladat eredményei is kerülnek. A számítógéppel való kommunikáció implementálása az egyedüli „hardver-közeli” feladat, ehhez a VCPU speciális utasításokat biztosít.

1.3 Licenc

A *FlexSD firmware*, illetve a hozzá készült VCPU bővítés GPLv2 licencű. Ezért a *firmware* részeinek vagy egészének felhasználásakor ezen licenc szerint kell eljárni. Viszont ez a licenc nem vonatkozik a VCPU által futtatott programra! A programozó a saját, VCPU által futtatott programjának olyan licencet választhat, amit szeretne[4].

---

1 A ***Flexibe SD drive firmware (FlexSD fw)*** egy ***sd2iec firmware*** fork. (Az eredeti a https://sd2iec.de/ oldalon található.) A dokumentáció további részében a *firmware* kifejezés a **FlexSD firmware**-t jelenti.

2 VCPU: „Virtual CPU”. Ezen meghajtók nagy része olyan mikrovezérlőre (CPU-t, adat és programmemóriát, illetve perifériákat tartalmazó integrált áramkör, tulajdonképpen egy komplett számítógép) épül, ami csak a ROM-jában levő programot tudja futtatni. A RAM-ba csak adatok kerülhetnek! A VCPU a mikrovezérlő ROM-jában levő szoftveresen megvalósított CPU, ez hajtja végre a RAM-ba másolt programot.

3 A dokumentációban az **SD2IEC** név magára a hardverre vonatkozik, ezek megegyeznek az eredeti *sd2iec firmware* által támogatott meghajtók hardverével.

4 A készített programot szoftver elemzi és dolgozza fel, illetve nincs semmilyen védelmi mechanizmus ezen program „visszafejtése” ellen, ezzel a programozónak tisztában kell lennie.

2 A számítógép által használható parancsok

Az SD2IEC meghajtó programozásánál szükség van néhány olyan parancsra, aminek a segítségével a számítógép a VCPU-s funkciókat elérheti. Ezeket a parancsokat (a KERNAL segítségével) a megszokott módon, a meghajtó un. „parancs-csatornájába” töltve kell kiadni, a meghajtótól jövő válaszokat a „hibacsatornából” lehet kiolvasni. (A programozás megegyezik a hagyományos CBM meghajtókéval, a részletek megtalálhatóak az ismert dokumentációkban.)

A VCPU bővítés parancsai **„Zx”** formájúak. Ha ismeretlen VCPU-s parancsot kap a meghajtó, „`35,SYNTAX ERROR,00,00`” hibaüzenettel válaszol.

2.1 „ZI”: Információk kérése a VCPU bővítésről

A válasz 4 BYTE:

1. VCPU verzió, ez két részből áll: B5..0[5]: maga a verziószám (jelenleg `$1`[6]), B7..5: az eszköz busz-típusa (ez jelenleg `$2`: CBM SERIAL).
2. „Parancs-csatorna” mérete BYTE-okban, ez általában 120, ennél kisebb nem fordul elő. A gép a meghajtó „parancs-csatornájába” maximum ennyi BYTE-ot írhat!
3. „Hibacsatorna” mérete BYTE-okban, ez általában 100, ennél kisebb nem fordul elő. A meghajtótól a gép felé a „hibacsatornán” keresztül maximum ennyi BYTE küldhető!
4. Adatpufferek száma, ez 6 vagy több (maximum 15). A meghajtó a fájlok kezeléséhez ennyi darab 256 BYTE-os puffert tart fenn. Ez egyben a VCPU memória-méretét is megadja, ennyi × 256 BYTE a használható programmemória. (6×256 = 1.5 KBYTE, 15×256 = 3.75 KBYTE.)

A visszaadott VCPU verzió jelenleg `$41`. (`%010` + `%00001`.) Ha a meghajtó nem ismeri a VCPU bővítést, egy „`30,SYNTAX ERROR,00,00`” hibaüzenettel válaszol. Ebben a válaszban az első BYTE mindig `$33` (a „3” ASCII kódja). A VCPU verzió a későbbiekben sem fog `$33` értéket felvenni, ezzel egyszerűen tesztelhető a meghajtó *firmware*-ben a támogatás megléte. (A VCPU elérhetőségének a tesztelésére ez a parancs az ajánlott. A meghajtó (normál / hosszú) verziójában található karakterek nem szükségszerűen utalnak a VCPU támogatásra, azok a későbbiekben változhatnak!)

5 Egy BYTE-on belül a bitek jelölése „**B**” + a bit száma. Több bit esetén a tartomány van megadva! (**B5..0**) A bitek számozása a szokásos: a legkisebb helyiértékű bit a 0-s (**B0**), a legnagyobb a 7-es (**B7**).

6 A dokumentációban a számrendszerek jelölése a 6502-s *assemblerek* szokásos jelölésmódja: a 16-os számrendszert „**$**”, a 2-st „**%**” prefix jelzi. A prefix nélküli számok 10-es számrendszerben értendőek.

2.2 „ZB”: Az adatpufferek állapotának lekérdezése

A válasz annyiszor 2 BYTE, ahány adatpuffer használható az eszközben:

1. Adatpuffer foglaltságjelző, ennek a 0-s bitje (B0) ha `%1`, akkor puffer foglalt, `%0` esetén szabad. A többi bit a meghajtó *firmware*-ben használatban lehet, a pontos jelentésük itt érdektelen. (A B0-n kívüli többi bit jelentése a későbbiekben változhat, emiatt azok állapotát nem szabad figyelembe venni!)
2. Az adatpufferhez tartozó csatornaszám (az OPEN utasítás 3. paramétere: „másodlagos cím”), amivel a gép hivatkozik rá. Ez csak akkor érvényes, ha a puffer foglalt (előző BYTE B0 = `%1`)! Az értéke 0..14 között van, ha az adott adatpuffer normál fájlkezelésre van használva. A meghajtó *firmware* speciális használatra is lefoglal(hat) adatpuffer(eke)t, a hozzá tartozó csatornaszám ekkor ezen tartományon kívüli értéket vesz fel.

A válaszban a fenti két BYTE annyiszor ismétlődik, ahány adatpufferrel a meghajtó rendelkezik.

2.3 „ZR”+ADDRLO+ADDRHI+LENGTH: Az adatpufferek olvasása, mint memória

Ez az 15x1-es meghajtók „`M-R`” parancsának felel meg, de itt az adatpufferek területe olvasható csak, nem az SD2IEC eszköz teljes memóriája. Az (egy paranccsal) olvasható memória mérete maximum a „hibacsatorna” mérete lehet, ennél több BYTE olvasása hibát ad vissza! („`35,SYNTAX ERROR,01,00`”) A „`LENGTH`” paramétert elhagyva 1 db. BYTE lesz visszaadva. Ha az adatokat a kezelhető memóriatartományon kívülről kellene olvasni, szintén hiba lesz a válasz! („`35,SYNTAX ERROR,02,00`”)

2.4 „ZW”+ADDRLO+ADDRHI+BYTE1+…: Az adatpufferek írása, mint memória

Az 15x1-es meghajtók „`M-W`” parancsához hasonló. A megadott címtől kezdődően annyi BYTE lesz a memóriába írva, amennyi BYTE a csomagban szerepel. (A parancsban itt nem szerepel külön a BYTE-szám, ellentétben az „`M-W`” paranccsal!) Az egyszerre beírható BYTE-ok mennyisége a „parancs-csatorna” méretétől függ, abba a teljes memóriaíró parancsnak adatokkal együtt bele kell férnie! Amennyiben a parancs a kezelhető memóriatartományon kívülre írna, hiba lesz visszaadva. („`35,SYNTAX ERROR,03,00`”) Ha a memória írása megtörtént, „`00, OK,00,00`” lesz a válasz.

2.5 „ZE”+ADDRLO+ADDRHI+…: Program végrehajtása a VCPU segítségével

A „ZE” parancs utáni paraméterekkel nem csak az indítási cím adható meg, hanem a VCPU összes regisztere beállítható. Ha nincs semmilyen paraméter megadva (a cím se), akkor a futtatás az aktuális állapottal és pozíciótól fog folytatódni. A paraméterek a „ZC” parancsnál találhatók meg. A válasz (és a lekérdezés lehetősége) a futtatott programtól függ, ami befejezéskor bármit visszaadhat.

FIGYELEM: A **„ZE”** parancs kiadása és a VCPU-s program elindulása között eltelő idő nem garantált hosszúságú, a meghajtó típusától és egyéb körülményektől függően változhat! Emiatt ajánlott valamilyen szinkronizációt beépíteni a programba, aminek a segítségével a számítógép „értesül” a meghajtóban levő program elindulásáról. (Például a soros busz valamelyik vezetékén a meghajtó generál egy impulzust, és ezt a számítógép megvárja.)

2.6 „ZC”: VCPU állapot lekérdezése

A parancs visszaadja a VCPU (aktuális) állapotát:

- **PCL** + **PCH**: két BYTE a PC aktuális pozíciója
- **A**: Akkumulátor
- **X**: X indexregiszter
- **Y**: Y indexregiszter
- **SR**: állapotregiszter
- **SP**: veremmutató
- **SPH**: veremmutató felső bitek (B15..8)
- **ZPH**: nulláslap felső bitek (B15..8)
- **INT**: megszakítás kódja
- **FUNCT**: hívott funkció kódja
- **LASTOP**: utoljára végrehajtott utasítás kódja

A „ZE” parancsban a paraméterek a **PCL**-től a **ZPH**-ig bezárólag ezek az adatok, ebben a sorrendben. Ha nincs cím se megadva, akkor az itt lekérdezhető címtől fog a futás elindulni. Az **INT** / **FUNCT** / **LASTOP** adatok hibakereséshez használhatóak.

3 A VCPU

3.1 Főbb különbségek

A VCPU egy szoftveresen megvalósított 6502 emulátorból, és a hozzá kapcsolt (szintén emulált) hardverből áll. A 6502 emuláció nem teljes, vannak hiányok, illetve plusz funkciók is:

- A cikluspontos sebességű végrehajtás nem volt cél; a jelenlegi implementáció egy, körülbelül 1 MHz-es 6502 sebességén futtatja a programot. (Bizonyos utasítások gyorsabbak, mások lassabbak.)
- Az eredeti MOS 6502 „nem-dokumentált” utasításai nincsenek támogatva! Ezen utasításkódok futtatásakor hibakód kíséretében megszakad a végrehajtás. (Viszont az extra utasítások ezen utasításkódok egy részére lettek kiosztva, azok a nekik szánt funkciót hajtják végre.)
- A BCD mód nincs implementálva! A `CLD` utasítás `NOP`-nak felel meg, a `SED` „illegális”.
- A hagyományos megszakításkezelés hiányzik. A `SEI` utasítás `NOP`, a `CLI` „illegális”. Az `RTI` utasítás viszont implementált, a szokásos módon használható.
- A program végrehajtás csak a RAM-nak kijelölt területről lehetséges! Ez a memória az emulált 6502 `$0000..$05FF` címtartománya 6 adatpuffer esetén, `$0000..$0EFF` 15 puffernél. Ezen tartományon kívülre ugrás esetén a végrehajtás hibakód kíséretében megszakad!
- A `JMP ($xxFF)` utasítás a vártnak megfelelően működik, az eredeti 6502 hibás működése nincs implementálva. Az indirekt címnek, ahol az ugrás valódi célcíme található, szintén a RAM-területen kell elhelyezkednie!
- A `BRK` utasítás a rendszerhívásokhoz van használva, de ezeket nem a CPU emuláció hajtja végre. Emiatt a `BRK` verem-műveletei nincsenek emulálva.

3.2 A VCPU kibővített funkciói

Ezen funkciókat az eredeti 6502 nem támogatja!

- **`SPH`** regiszter: az eredeti 6502-s veremmutató (**`SP`**) 8 bites, a felső 8 bit ott mindig `$01`. A VCPU-nál ezt a felső 8 bitet az **`SPH`** regiszter tárolja, így a verem másik memórialapra helyezhető. A 8 bites **`SP`** esetleges átfordulása nem módosítja az **`SPH`**-t!
- **`ZPH`** regiszter: az eredeti 6502-s nulláslap mindig a `$0000..$00FF` területen helyezkedik el. A VCPU-nál a cím felső 8 bitje állítható, így a nulláslap másik memórialapra helyezhető. De ez csak a nulláslapot használó címzésmódokat érinti! Átállított **`ZPH`** esetén a direkt címes utasítások (például: `LDA $0080`) mindig a valódi címmel[7] dolgoznak!
- A CBM SERIAL, azaz a soros busz kezelésére speciális „mikró” utasítások használhatóak, amikkel „elemi lépésenként” leprogramozható bármilyen adatátvitel. Ezen utasítások többségének „beállított” a futásideje, így az időkritikus, időzítéses adatátvitelek is megoldhatóak.

7 Erre különösen az *Y-indexelt* címzésmódnál kell odafigyelni, ugyanis az utasításoknak általában nincs `$zp,Y` címzésmódja! A fordítók viszont hibaüzenet nélkül `$qwer,Y` címzésmódot fordítanak ilyenkor, ami – áthelyezett nulláslap esetén – nem a várt címet használja.

3.3 Az emulált 6502 memóriatérképe

- `$0000..$05FF` / `$0000..$0EFF`: A kezelt fájlok memóriapufferei, illetve RAM terület a program számára
- `$FC00..$FCFF`: A meghajtó „parancs-csatornája” ezen a területen írható/olvasható, de csak annyi BYTE, amilyen méretű a csatorna
- `$FD00..$FDFF`: A meghajtó „hibacsatornája” ezen a területen írható/olvasható, de csak annyi BYTE, amilyen méretű a csatorna
- `$FE00..$FEFF`: I/O terület, jelenlegi hossza 16 BYTE

3.4 Az I/O regiszterek

- `$FE00`: VCPU verziószám, csak olvasható. (Jelenleg `$41`, ugyanaz az érték, amit a „`ZI`” parancs válaszában szerepel.)
- `$FE01`: Az SD2IEC eszköz jelenlegi eszközszáma (8..11), csak olvasható.
- `$FE02`: Az SD2IEC nyomógombok és LED-ek állapota, írható/olvasható. B0 = „BUSY LED”, B1 = „DIRTY LED” állapota. 1: a LED világít. B6 = „PREV BUTTON”, B7 = „NEXT BUTTON” állapota. 1: a gomb nyomva.
- `$FE03..$FE04`: Diagnosztika, funkciójuk a későbbiekben változhat, nem használható!
- `$FE05..$FE09`: nem használt, `$00`
- `$FE0A`: ATN / SRQ vonalak vezérlése, írható/olvasható. B7 = ATN, B6 = SRQ. Írásnál a vonalak hajtása kapcsolható be/ki, a regiszter olvasásakor ez a beállított állapot olvasódik vissza, nem a vonalak aktuális állapota!
- `$FE0B`: ATN / SRQ vonalak aktuális állapota, csak olvasható! B7 = ATN, B6 = SRQ.
- `$FE0C`: CLK / DAT vonalak vezérlése, írható/olvasható. B1 = DAT, B0 = CLK. Írásnál a vonalak hajtása kapcsolható be/ki, a regiszter olvasásakor ez a beállított állapot olvasódik vissza, nem a vonalak aktuális állapota!
- `$FE0D`: CLK / DAT vonalak aktuális állapota, csak olvasható! B7 = DAT, B6 = CLK.
- `$FE0E`: CLK vonal aktuális állapota, csak olvasható! B7 = CLK.
- `$FE0E`: CLK vonal vezérlése, csak írható! B0 = CLK.
- `$FE0F`: DAT vonal aktuális állapota, csak olvasható! B7 = DAT.
- `$FE0F`: DAT vonal vezérlése, csak írható! B0 = DAT.

Az `$FE02` címen a készüléken található LED-ek vezérelhetőek. A két LED nem mindegyik eszközön található meg! Van olyan SD2IEC hardver, amin csak egy LED van, ott az a „BUSY LED” bitjén keresztül kapcsolható. A LED-ek egyszerűen programozhatóak, viszont a rendszerhívások (fájl nyitása/zárása, stb.) a szokásos módon kapcsolgathatják őket, emiatt a beállított állapotok nem biztos, hogy megmaradnak. („Normál használatnál” célszerű a meghajtó *firmware*-re hagyni a vezérlésüket.) Hiba esetén a megszokott „villogó LED” nem működik a VCPU-s program futtatása alatt!

Ugyanezen az `$FE02` címen elérhető még a két nyomógomb állapota is. A VCPU-s programfutás alatt a nyomógombok eredeti funkciói nem működnek! Van olyan SD2IEC hardver,

amibe ezek a nyomógombok nincsenek beépítve! Emiatt ezen gombok használata nem javasolt. Az esetleges lemezkép-cserére van lehetőség a program oldaláról, de ezt célszerű automatikusra, a felhasználó beavatkozását nem igénylő módon megvalósítani.

Az `$FE0A..$FE0F` tartományban a soros busz vonalainak a vezérlő regiszterei találhatók. Ezen regiszterek a későbbiekben sem lesznek bővítve, a nem használt bitek olvasáskor garantáltan `%0` értéket adnak vissza, íráskor a nem használt bitekre tetszőleges érték írható! A soros busz kezelése ezeken a regisztereken keresztül megvalósítható de nem ajánlott, erre a feladatra a VCPU utasításkészletében külön utasítások találhatóak.

4 A CBM soros busz

4.1 Hardver

A soros busz vezetékei használaton kívül logikai magas szinten vannak. Ezt a magas szintet bármelyik buszon levő eszköz (számítógép / periféria) alacsony szintre tudja kapcsolni. Viszont magas szint a busz bármelyik vezetékén csak akkor tud kialakulni, ha az összes eszköz kikapcsolja az alacsony szintre hajtást! Emiatt a leírásban a „vonal hajtása” mindig azt jelenti, hogy az adott eszköz alacsony szintre húzza a vonalat, a „vonal elengedése” meg azt, hogy az alacsony szintre hajtást kikapcsolja. Ezen utóbbi esetben az adott vonalon csak akkor fog (tud) magas szint kialakulni, ha az összes eszköz „elengedi” azt! (Magas szintet „kényszeríteni” a vonalakra egyik eszköz sem tud.)

4.2 Szoftver

A soros busz vonalait a szoftverek valamilyen periféria regiszterének a bitjeiként tudják kezelni. Az adott vonal „jelszintjét” a hozzá tartozó periféria-bit értéke mutatja. A vonal „hajtása” / „elengedése” a periféria egy másik bitjének a kapcsolgatásával vezérelhető. Az SD2IEC oldalán a periféria adott bitje mindig a vonalak aktuális jelszintjét mutatja (alacsony vonal esetén `%0`, magas esetén `%1`), a kimenet vezérlése is ugyanígy történik (`%0` húzza alacsonyra, `%1` elengedi).

4.3 Vonalak

A soros busznak négy, programozás szempontjából érdekes vezetéke van, ezek (és az eredeti funkcióik) a következők:

- SRQ: Service ReQuest: kiszolgálás kérése. Nincs használva. Ezt a vezetéket a perifériák hajtanák, a számítógépen bemenet. Az 1541-ben ez a jel nincs bekötve. A VIC20-ban az egyik VIA fogadja, a CPU a valódi szintjét nem tudja olvasni, csak a változására megszakítás generálható. A C64-ben a működése ugyanez, az egyik CIA fogadja, megszakítás generálható vele. A C16 / C116 / plus/4 gépekben nincs bekötve. C128-on a „gyors soros busz” üzemmódban van szerepe, ezt a meghajtó *firmware* egyelőre nem támogatja. Az SRQ vonal vezérlése a VCPU segítségével jelenleg nincs implementálva! Az SD2IEC hardverek egy részén – mivel nincs használatban – a vonal vezérlése nincs is megvalósítva!
- ATN: ATtentioN: címzésre figyelmeztetés. Ezt a vezetéket a számítógép hajtja, a perifériákon bemenet. Alacsony szintjénél az eszközök „megszólítása” zajlik, a számítógép ilyenkor választja ki azt a perifériát, akivel kommunikálni akar. Az 1541-ben ez csak bemenet, a változása a DAT hajtás bekapcsolását eredményezi. Az SD2IEC – az alap CBM perifériáktól eltérően – tudja hajtani, a VCPU oldaláról vezérelhető. A VIC20, C64, C16 / C116 / plus/4, illetve a C128 esetén a vezeték csak kimenet, a vonal állapota nem olvasható vissza! Az SD2IEC általi vezérelhetőség akkor lehet hasznos, ha egy másik perifériával történik a kommunikáció közvetlenül, a számítógép kikerülésével.
- CLK: CLocK: az adatátvitel „órajele”, ennek a segítségével vannak az átvitt bitek szinkronizálva. A számítógép és a perifériák is tudják hajtani, illetve figyelni.
- DAT: DATa: az éppen átvitt adatbit. A számítógép és a perifériák is tudják hajtani / figyelni.

5 VCPU utasításkészlet

A VCPU utasításait 3 csoportra lehet bontani: az eredeti 6502-es utasítások, a bővített funkciók utasításai, illetve a soros busz kezeléséhez tartozó „mikró” utasítások. Az első csoportba tartozó eredeti utasítások leírása megtalálható a 6502 programozásával foglalkozó ismert dokumentumokban.

5.1 A bővített funkciók utasításai

Mnemonik	Op-kód	Regiszterek[8]	Leírás
TYSPH	$D4	SPH	Transfer Y to SPH register
TSPHY	$F4	Y,N,Z	Transfer SPH to Y register
LDSPH #$xx	$22 $xx	SPH	LoaD SPH register

- TYSPH: az **SPH** regiszterbe mozgatja az **Y** regiszter tartalmát
- TSPHY: az **Y** regiszterbe mozgatja az **SPH** regiszter tartalmát
- LDSPH #$xx: az **SPH** regiszterbe a megadott értéket tölti

TYZPH	$44	ZPH	Transfer Y to ZPH register
TZPHY	$54	Y,N,Z	Transfer ZPH to Y register
LDZPH #$xx	$02 $xx	ZPH	LoaD ZPH register

- TYZPH: a **ZPH** regiszterbe mozgatja az **Y** regiszter tartalmát
- TZPHY: az **Y** regiszterbe mozgatja a **ZPH** regiszter tartalmát
- LDZPH #$xx: a **ZPH** regiszterbe a megadott értéket tölti

Az **SPH** / **ZPH** regiszter csak a RAM területére mutathat! Amennyiben olyan érték kerülne ezen regiszterekbe, ami nem a memóriában van, akkor a programfutás hibakód kíséretében megszakad.

BTASC	$E2	A,X,Y	Binary To ASCii conversion

- BTASC: a VCPU nem támogatja a BCD üzemmódot, de időnként szükség van egy-egy szám decimális megfelelőjére. A BTASC az akkumulátor tartalmát alakítja át 000..255 karakterhármassá. Az **Y**:**X**:**A** regiszterek sorrendben a decimális érték számjegyeinek a karaktereit tartalmazzák. A státuszregiszter nem változik!

8 Az **N**, **V**, **B**, **D**, **I**, **Z**, **C** betűk a státusz-regiszter bitjeit, az **A**, **X**, **Y**, **SP**, **SPH**, **ZPH** a CPU adott regiszterét jelölik.

5.2 A soros busz kezeléséhez tartozó „mikró” utasítások

Ezen utasítások többsége nem módosítja a státuszregiszter bitjeit! A végrehajtási idők rögzítettek, a „mértékegység” az 1541-ben levő 1 MHz-es 6502 egy órajelciklusa. (Egy 6502 órajelciklus ideje lesz az „1T”.)

Mnemonik	Op-kód	Idő	Regiszterek	Leírás
UWATL	$43	1.5T+	–	Wait for serial ATN line Low
UWATH	$53	1.5T+	–	Wait for serial ATN line High
UWCKL	$23	1.5T+	–	Wait for serial CLK line Low
UWCKH	$33	1.5T+	–	Wait for serial CLK line High
UWDTL	$03	1.5T+	–	Wait for serial DAT line Low
UWDTH	$13	1.5T+	–	Wait for serial DAT line High

- UWATL: A soros busz ATN vezetékének alacsony szintjére vár
- UWATH: A soros busz ATN vezetékének magas szintjére vár
- UWCKL: A soros busz CLK vezetékének alacsony szintjére vár
- UWCKH: A soros busz CLK vezetékének magas szintjére vár
- UWDTL: A soros busz DAT vezetékének alacsony szintjére vár
- UWDTH: A soros busz DAT vezetékének magas szintjére vár

Ezen utasítások a soros busz megfelelő vonalának a kívánt szintjére várakoznak. Ha a vonal szintje az utasítás végrehajtása előtt már olyan amire várna, akkor a futásidő 1.5T. A várakozást a vonal szintjének a beállása után kevesebb mint 1T idő alatt befejezi, és folytatódik a végrehajtás.

USATL	$C3	1.5T	–	Set serial ATN line to Low
USATH	$D3	1.5T	–	Set serial ATN line to HighZ
USCKL	$A3	1.5T	–	Set serial CLK line to Low
USCKH	$B3	1.5T	–	Set serial CLK line to HighZ
USDTL	$83	1.5T	–	Set serial DAT line to Low
USDTH	$93	1.5T	–	Set serial DAT line to HighZ

- USATL: A soros busz ATN vezetékét alacsonyra kapcsolja
- USATH: A soros busz ATN vezetékét elengedi
- USCKL: A soros busz CLK vezetékét alacsonyra kapcsolja
- USCKH: A soros busz CLK vezetékét elengedi
- USDTL: A soros busz DAT vezetékét alacsonyra kapcsolja
- USDTH: A soros busz DAT vezetékét elengedi

Ezen utasítások a soros busz megfelelő vonalait vezérlik. A vezeték „elengedése” csak az SD2IEC oldaláról történik meg, más eszköz ettől még alacsony szinten tarthatja!

UCLDL	$0B	1.5T	-	Set serial CLK to Low / DAT to Low
UCLDH	$1B	1.5T	-	Set serial CLK to Low / DAT to HighZ
UCHDL	$2B	1.5T	-	Set serial CLK to HighZ / DAT to Low
UCHDH	$3B	1.5T	-	Set serial CLK to HighZ / DAT to HighZ

- UCLDL: A soros busz CLK és DAT vezetékét alacsonyra kapcsolja
- UCLDH: A soros busz CLK vezetékét alacsonyra kapcsolja, a DAT vezetékét elengedi
- UCHDL: A soros busz CLK vezetékét elengedi, a DAT vezetékét alacsonyra kapcsolja
- UCHDH: A soros busz CLK és DAT vezetékét elengedi

Ezekkel az utasításokkal a CLK és DAT vonalak egy lépésben vezérelhetők.

UATCK #$xx	$6B $xx	2T	-	Copy **A** register bit To CLK line
UCKTA #$xx	$7B $xx	2T	A	Copy CLK line To **A** register bit
UATDT #$xx	$4B $xx	2T	-	Copy **A** register bit To DAT line
UDTTA #$xx	$5B $xx	2T	A	Copy DAT line To **A** register bit

- UATCK #$xx: Az **A** regiszter kiválasztott bitje alapján hajtja / elengedi a CLK vonalat
- UCKTA #$xx: A CLK vonal állapotát bemásolja az **A** regiszter kiválasztott bitjébe
- UATDT #$xx: Az **A** regiszter kiválasztott bitje alapján hajtja / elengedi a DAT vonalat
- UDTTA #$xx: A DAT vonal állapotát bemásolja az **A** regiszter kiválasztott bitjébe

Az utasítások paramétere egy bitmaszk, amiben csak egyetlen bit %1 értékű! A CLK / DAT olvasásánál a kívánt vonal értéke bemásolódik az akkumulátor azon bitjébe, ahol az utasítás paraméterében szereplő bitmaszkban %1 szerepel. A vonalak írásánál az akkumulátor azon bitje alapján áll be a kívánt vonal hajtása / elengedése, amelyik bit a maszkban %1. Figyelem: a VCPU **SR** bitjei nem változnak akkor sem, amikor az akkumulátor értéke módosul!

`UATCD #$xx,#$yy`	`$8B $xx $yy`	`2.5T`	`-`	Copy **A** register bits To CLK / DAT
`UCDTA #$xx,#$yy`	`$9B $xx $yy`	`2.5T`	`A`	Copy CLK / DAT line To **A** register bits

- `UATCD #$xx,#$yy`: Az **A** regiszter kiválasztott bitjei alapján vezérli a CLK/DAT vonalakat
- `UCDTA #$xx,#$yy`: A CLK/DAT vonalak állapotát bemásolja az **A** regiszter kiválasztott bitjeibe

Az utasítások paramétere két bitmaszk, amikben csak egyetlen bit `%1` értékű! Az első paraméter a CLK vonalhoz tartozik, a második a DAT-hoz. Olvasásnál a CLK az akkumulátor azon bitjébe kerül, ami az első bitmaszkban `%1`, a DAT a második bitmaszk alapján ugyanígy. A vonalak írásánál az első bitmaszk alapján kiválasztott akkumulátor bit szerint kapcsolódik a CLK hajtása / elengedése, a második bitmaszk alapján meg a DAT vonalé. Figyelem: a VCPU **SR** bitjei nem változnak akkor sem, amikor az akkumulátor értéke módosul!

Ezen utasítások bitmaszkjainak csak olyan értékek megengedettek, amiben egy darab bit `%1`. (Azaz: `$01`, `$02`, `$04`, `$08`, `$10`, `$20`, `$40`, `$80`.) Az összes többi variáció végeredménye „nem definiált”. (Olvasáskor ha több bit is `%1`, a kiválasztott vonal az akkumulátor összes, `%1`-gyel jelölt bitjébe bekerül, ez valószínűleg nem fog a későbbiekben sem változni. Íráskor ha több bit is `%1`, akkor a hozzá tartozó akkumulátor-bitek valamilyen logikai kombinációja határozza meg a vonal állapotát, ami akár az SD2IEC hardveres verziójától is függhet, emiatt nem javasolt a „tiltott” bitmaszkok használata!) Az egyszerre két vonalat is olvasó / író utasításoknál a két bitmaszk nem lehet ugyanaz! (Olvasásnál „nem definiált”, hogy melyik vonal állapota lesz végül az akkumulátor kiválasztott bitjében. Íráskor mindkét vonal állapotát az akkumulátor ugyanazon bitje határozza meg, ez valószínűleg szintén nem fog változni.)

`ULBIT`	`$DB`	`2T`	`N,V,Z`	Lines BIT test

- `ULBIT`: A soros vonalak aktuális állapotát a státuszregiszter bitjeibe másolja

Az utasítás a soros vonalak aktuális állapotát másolja a státuszregiszter bitjeibe: az ATN a V, a CLK az N, a DAT a Z bitbe kerül. Az utasítást követő feltételes elágazások a vizsgálandó vonal állapota alapján ugranak. (`BVS` / `BVC`: ATN vonal magas / alacsony, `BMI` / `BPL`: CLK vonal magas / alacsony, `BEQ` / `BNE`: DAT vonal magas / alacsony.)

USND1	$63	?T		Send **A** register to DAT, 1 bit mode
URCV1	$73	?T	A	Receive from DAT to **A** register, 1 bit mode
USND2	$E3	?T		Send **A** register to CLK/DAT, 2 bit mode
URCV2	$F3	?T	A	Receive from CLK/DAT to **A** register, 2 bit mode

- USND1: 1 bites szinkronizált adatküldés
- URCV1: 1 bites szinkronizált adatfogadás
- USND2: 2 bites szinkronizált adatküldés
- URCV2: 2 bites szinkronizált adatfogadás

Az „1 bites” utasítások a CLK / DAT vonalak segítségével egy egész BYTE szinkronizált küldését / fogadását valósítják meg. A szinkronizálás a CLK vonal segítségével valósul meg, ezt a számítógép kapcsolgatja, ennek a változására reagál a meghajtó. A futásidő emiatt nem meghatározható, a számítógép tempójától függ:

- Az USND1 utasítás adatküldés. Első lépésben a CLK vonal hajtását kikapcsolja, majd vár ugyanennek a vonalnak a magas állapotára. Ha a CLK magas, a DAT vonalra másolja az **A** regiszter 0-s bitjét. Ezután a CLK vonal alacsony szintjére vár. Ha a számítógép átvette a B0-t, alacsonyra kapcsolja a CLK vonalat. Erre a meghajtó az **A** regiszter 1-es bitjét másolja a DAT vonalra, majd CLK magasra vár. Ha a számítógép átvette a B1-et, a CLK vonalat elengedi. Erre a meghajtó a következő bitet rakja a DAT vonalra, és így tovább. A BYTE végén a számítógép a CLK vonalat alacsonyra kapcsolja, erre a meghajtó kirakja a 7. bitet az **A** regiszterből a DAT vonalra, majd az utasítás befejeződik, jön a következő utasítás végrehajtása.
- Az URCV1 utasítás adatfogadás. Első lépésben a CLK / DAT vonal hajtását kikapcsolja, majd vár egy CLK alacsony állapotra. A számítógép a DAT vonalra a küldendő adat B0-t másolja, majd alacsonyra kapcsolja a CLK vonalat. Erre a meghajtó beolvassa a DAT vonal állapotát az **A** regiszter 0-s bitjébe, majd vár a CLK magas állapotra. A gép beállítja a DAT vonalra a B1-es bitet, majd a CLK vonalat magasra kapcsolja. Erre a meghajtó beolvassa a DAT vonal állapotát az **A** regiszter 1-es bitjébe, és így tovább. A BYTE végén a számítógép a 7. bitet másolja a DAT vonalra, illetve a CLK vonalat elengedi, ekkor a meghajtó beolvassa az **A** regiszter B7-be a DAT-ot, majd az utasítás befejeződik, jön a következő végrehajtása.

A „2 bites” utasítások az ATN / CLK / DAT vonalak segítségével egy egész BYTE szinkronizált küldését / fogadását valósítják meg. A szinkronizálás az ATN vonal segítségével valósul meg, ezt a számítógép kapcsolgatja, ennek a változására reagál a meghajtó. A futásidő emiatt nem meghatározható, a számítógép tempójától függ:

- Az USND2 utasítás adatküldés. Első lépésben az ATN vonal hajtását kikapcsolja, majd vár ugyanennek a vonalnak a magas állapotára. Ha az ATN magas, a DAT vonalra másolja az **A** regiszter 1-es, a CLK vonalra meg a 0-s bitjét. Ezután az ATN vonal alacsony szintjére vár. Ha a számítógép átvette a B1/B0-t, alacsonyra kapcsolja az ATN vonalat. Erre a meghajtó az **A** regiszter 3-as bitjét másolja a DAT, a 2-s bitjét a CLK vonalra, majd ATN magasra vár. Ha

a számítógép átvette a B3/B2-t, az ATN vonalat elengedi. Erre a meghajtó a következő bitpárt rakja a DAT/CLK vonalakra, és így tovább. A BYTE végén a számítógép az ATN vonalat alacsonyra kapcsolja, erre a meghajtó kirakja a 6. / 7. bitet az **A** regiszterből a CLK/DAT vonalakra, majd az utasítás befejeződik, jön a következő utasítás végrehajtása.

- Az `URCV2` utasítás adatfogadás. Első lépésben az ATN / CLK / DAT vonal hajtását kikapcsolja, majd vár egy ATN alacsony állapotra. A számítógép a DAT-ra a küldendő adat B0-t, a CLK-ra a B1-et másolja, majd alacsonyra kapcsolja az ATN vonalat. Erre a meghajtó beolvassa a DAT vonal állapotát az **A** regiszter 0-s, a CLK vonalét meg az 1-es bitjébe, majd vár az ATN magas állapotra. A gép beállítja a DAT-ra a B2-t / a CLK-ra a B3-at, majd az ATN vonalat magasra kapcsolja. Erre a meghajtó beolvassa a DAT állapotát az **A** regiszter 2-s bitjébe, a CLK állapotát a 3-asba, és így tovább. A BYTE végén a számítógép a 6 / 7. bitet másolja a DAT / CLK vonalra, illetve az ATN vonalat elengedi, ekkor a meghajtó beolvassa az **A** regiszter B6-ba a DAT / B7-be a CLK állapotát, majd az utasítás befejeződik, jön a következő végrehajtása.

`UINDB $offs`	`$AB $of`	`1.5T/2T`	`X,Y`	INcrement **X**, Decrement **Y**, Branch if no cy.
`UDEDB $offs`	`$BB $of`	`1.5T/2T`	`X,Y`	DEcrement **X**, Decrement **Y**, Branch if no cy.

- `UINDB $offs`: **X** növelése, **Y** csökkentése, majd ugrás, ha **Y** nem lett még `$FF`
- `UDEDB $offs`: **X** csökkentése, **Y** csökkentése, majd ugrás, ha **Y** nem lett még `$FF`

Az utasítások növelik vagy csökkentik az **X** regiszter értékét. Majd az **Y** regiszterét csökkentik, ami ha átfordul `$00`-ról `$FF`-re, akkor a következő utasítással folytatódik a végrehajtás. Ekkor a futásidő 1.5T. Ha az **Y** nem lett még `$FF`, akkor az utasításkód utáni paraméter által kijelölt címre ugrik a végrehajtás, ekkor a teljes végrehajtási idő 2T. (Az ugrás címe ugyanúgy számítódik, mint a többi 6502-s feltételes ugrásnál, a PC aktuális értékéhez adódik hozzá a paraméter előjeles számként. Így -128..+127 tartományba lehet ugrani a segítségével.) Az esetleges 256 BYTE-os laphatár átlépése nem módosítja a végrehajtási időt. Figyelem: a VCPU **SR** bitjei nem változnak az utasítások végrehajtásakor!

`UDEL1`	`$EA`	`1T`	–	1T DELay
`UDELY #$xx`	`$FB $xx`	`1.5T+`	–	DELaY

- `UDEL1`: 1T idejű késleltetés
- `UDELY #$xx`: 1.5T + `$xx` × 0.5T idejű késleltetés

Az `UDEL1` utasítás 1T idejű késleltetés. (Ez az eredeti 6502 `NOP` kódja, a futásidő 1T.) Az `UDELY` egy olyan késleltető utasítás, aminek az idejét a paraméterében megadott fix érték adja meg. A legkisebb idő 1.5T. Ehhez az időhöz jön hozzá a paraméterként megadott érték × 0.5T.

Ezen utasítások még két adatmozgató utasítással egészülnek ki:

`LDA $zp,X`	`$B5 $zp`	`2T`	`A,N,Z`	LoaD **A**ccumulator from zero page **X** indexed
`STA $zp,X`	`$95 $zp`	`2T`	`-`	Store **A**ccumulator to zero page **X** indexed

Ezek az eredeti 6502-s **X** regiszter indexelt nulláslapos olvasó / író utasítások, csak a futásidejük 2T. Mivel a VCPU-ban a nulláslap bárhova áthelyezhető (a memórián belül), így az egész memóriából bárhonnan tudnak olvasni, illetve bárhova lehet velük írni.

5.3 A meghajtó *firmware* különböző funkcióinak hívása

Mnemonik	Op-kód	Regiszterek	Leírás
`BREAK #$xx`	`$00 $xx`	`A,X,Y`	System Call

- `BREAK #$xx`: az $xx számú rendszerfunkció meghívása

Az utasítás a 6502 eredeti `BRK` utasítása. Az utasítást és az utána következő paraméter BYTE-ot nem a VCPU dolgozza fel, hanem közvetlenül a meghajtó *firmware*. A VCPU aktuális állapota mentődik, majd a kívánt rendszerhívás megtörténik.

6 A rendszerhívások

A `BREAK #$xx` utasítással a meghajtó *firmware* különböző funkciói érhetők el. A paraméterként megadott szám a hívandó rendszerhívás száma. Ha a funkció a CPU emuláció leállítását kéri, akkor a programfutás megszakad. Más feladatnál, annak a végrehajtása után a 6502-s kód emulált futtatása folytatódni fog a `BRK` + paraméter BYTE utáni címtől. A rendszerhívás az esetleges visszaadandó adatait a VCPU **A** / **X** / **Y** regisztereibe írja bele, emiatt ezen regiszterek tartalma nem őrződik meg! A státuszregiszter bitjei nem állnak be egyik regiszter alapján sem, a visszakapott adatokat ellenőrizni kell!

6.1 A rendszerhívások funkció-kódjai

Kód	Hivatkozási név	Leírás
$00	SYSCALL_EXIT_OK	Kiszállás „00, OK,00,00” üzenettel
$01	SYSCALL_EXIT_SETERROR	Kiszállás a kiválasztott üzenettel
$02	SYSCALL_EXIT_FILLEDERROR	Kiszállás úgy, hogy a „hibacsatorna” fel van töltve
$03	SYSCALL_EXIT_REMAIN	Kiszállás úgy, hogy a „hibacsatorna” nem módosul
$11	SYSCALL_DISABLEATNIRQ	Tiltja az ATN aktív esetén történő VCPU kiszállást
$12	SYSCALL_ENABLEATNIRQ	Engedélyezi az ATN aktív esetén történő VCPU kiszállást
$13	SYSCALL_SETFATPARAMS	FAT fájl kezeléséhez paraméterek beállítása
$21	SYSCALL_DIRECTCOMMAND	A „parancs-csatornába” levő parancs végrehajtása
$22	SYSCALL_DIRECTCOMMAND_MEM	A memóriában összeállított parancs „parancs-csatornába” történő másolása és végrehajtása
$23	SYSCALL_OPEN	Fájl megnyitása (név a „parancs-csatornában”)
$24	SYSCALL_OPEN_MEM	Fájl megnyitása (név a memóriában)
$25	SYSCALL_CLOSE	Fájl lezárása
$26	SYSCALL_CLOSEALL	Minden nyitott fájl / csatorna lezárása
$27	SYSCALL_REFILLBUFFER	Puffer újratöltése olvasáskor / kiürítése íráskor
$28	SYSCALL_GETCHANNELPARAMS	Csatorna paraméterek lekérdezése
$31	SYSCALL_CHANGEDISK	Lemezcsere kérelem

6.2 Rendszerhívások leírásai

- $00: SYSCALL_EXIT_OK: A VCPU-s programfutás befejezése. A „hibacsatorna” tartalma a „00, OK,00,00” üzenet lesz, a számítógép innentől a megszokott módon tudja a perifériát kezelni.
- $01: SYSCALL_EXIT_SETERROR: A VCPU-s programfutás befejezése. A „hibacsatornába” a kért hiba lesz beállítva! A rendszerhívás előtt az **A** regiszterbe a hibakód számát, az **X** / **Y** regiszterekbe a sáv / szektor számát kell tölteni, ez alapján fog a hiba szövege beállni. Csak az egyébként használt hibaszámok megengedettek!
- $02: SYSCALL_EXIT_FILLEDERROR: A VCPU-s programfutás befejezése. A „hibacsatorna” tartalmát előtte a kívánt adatokkal fel kell tölteni, majd az **X** regiszterbe a bekerült BYTE-ok számát be kell állítani. Utána hívható a funkció.
- $03: SYSCALL_EXIT_REMAIN: A VCPU-s programfutás befejezése. A „hibacsatorna” tartalma változatlan marad! (Amennyiben egy előzőleg hívott funkció hibával tért vissza, a VCPU-s programfutást így befejezve a számítógép a szokásos módon lekérdezheti a hibát.)

- `$11`: `SYSCALL_DISABLEATNIRQ`: Az alap működés során ha az ATN vezeték aktív lesz, a VCPU-s programvégrehajtás megszakad. Amennyiben szükség van az ATN használatára, ezen funkció hívásával ez a viselkedés kikapcsolható.
- `$12`: `SYSCALL_ENABLEATNIRQ`: Visszakapcsolja az ATN vezeték aktív állapotára történő VCPU-s programvégrehajtás megszakítást.
- `$13`: `SYSCALL_SETFATPARAMS`: A meghajtó *firmware* a FAT fájlrendszerek esetén is 254 BYTE-os darabokban olvassa / írja a fájlokat, ahogyan ezt a lemezképeken belül is teszi. Ezzel a rendszerhívással ezt a viselkedést lehet átállítani. Hívás előtt az **X** regiszterbe kell az adatpufferen belüli kezdőpozíciót rakni (ez alapesetben 2, innentől kezdődnek a betöltött adatok), az **Y** regiszterbe meg az egy lépésben beolvasandó BYTE-számot. (Ez alapesetben 254.) A 0 érték 256 BYTE-ot jelent! A rendszerhívás után az **X** / **Y** regiszterekben a beállított értékek szerepelnek. Ha a kezdőpozíció + BYTE-szám nagyobb mint az adatpuffer mérete (256), a beállítás nem történik meg! Viszont visszatérés után a regiszterekben az aktuálisan beállított értékek szerepelnek, ezért egy „hibás” paraméteres hívással lekérdezhető a jelenlegi beállítás.
- `$21`: `SYSCALL_DIRECTCOMMAND`: A meghajtó számára a számítógép a „parancs-csatornán” keresztül tudja közölni a direkt fájlkezelésen kívüli feladatokat. (Fájl törlése, könyvtárváltás, stb.) A VCPU ugyanezen módon tud parancsot végrehajtatni: a „parancs-csatorna” mint memória be van „fűzve” a 6502 memóriájába. Ide a parancs BYTE-jait közvetlenül be kell írni. A „parancs-csatornába” került BYTE-ok számát az **X** regiszterbe töltve kell ezt a rendszerhívást elindítani. A meghajtó *firmware* megpróbálja végrehajtani a parancsot, majd visszatér a VCPU-s programfuttatáshoz. A rendszerhívás után az **A** regiszterben a „hibacsatornába” került hiba kódja található, az **X** regiszter a bekerült BYTE-ok számát tartalmazza.
- `$22`: `SYSCALL_DIRECTCOMMAND_MEM`: A parancs megegyezik az előző (`$21`-es kódú) paranccsal, de előtte a VCPU memóriájából a „parancs-csatornába” másolja a kívánt BYTE-sorozatot. A funkció hívása előtt a memóriaterület kezdőcímét a **ZPH**:**Y** regiszterpárba kell beállítani, az **X** regiszter a másolandó BYTE-ok számát (és ezzel együtt a parancs hosszát) tartalmazza. A visszakapott paraméterek a `$21`-es parancsnál leírtakkal megegyeznek.
- `$23`: `SYSCALL_OPEN`: Fájl megnyitása. A hívás előtt a „parancs-csatornába” a megnyitandó fájl nevét és üzemmódját kell beírni, az **X** regiszterbe a fájlnév hosszát kell beállítani. Az **A** regiszterbe a kért csatorna száma kerül, ez 0..14 között lehet. Ezután kell ezt a rendszerhívást elindítani. A visszatérés után az **A** regiszterben a „hibacsatornába” került hiba kódja található, az **X** regiszter az ugyanide bekerült BYTE-ok számát tartalmazza. Az **Y** regiszter annak az adatpuffernek a *sorszámát* tartalmazza, ami a fájlhoz hozzá lett rendelve. Ez az érték a VCPU memóriájának a felső 8 bitje, ha itt például `$02` van, akkor a puffer a memóriában a `$0200`..`$02FF` tartományban található! Ezen felül a „hibacsatorna” memóriaterületén (`$FDxx`) található még három adat: hol kezdődnek az adatpufferben a betöltött adatok (`$FD60`), melyik az utolsó, még használt BYTE (`$FD61`), illetve van-e még vissza adat a fájlból (`$FD62`).
- `$24`: `SYSCALL_OPEN_MEM`: A parancs megegyezik az előző (`$23`-as kódú) paranccsal, de előtte a fájlnevet a VCPU memóriájából a „parancs-csatornába” másolja. A funkció hívása előtt a memória kezdőcímét a **ZPH**:**Y** regiszterpárba kell beállítani, az **X** regiszter a másolandó BYTE-ok számát (és ezzel együtt a fájlnév hosszát) tartalmazza. Az **A** regiszterbe a kért csatorna száma kerül mint a `$23`-as parancsnál, a visszaadott paraméterek is megegyeznek vele.

- $25: SYSCALL_CLOSE: Fájl / csatorna lezárása. Az **A** regiszterbe az OPEN esetén megadott csatorna számát kell írni, majd a rendszerhívást el kell indítani. A végrehajtás után az **A** regiszter tartalma $00 ha sikeres volt a lezárás.
- $26: SYSCALL_CLOSEALL: Az összes megnyitott fájl / csatorna lezárása, paraméterek / visszaadott adatok nincsenek.
- $27: SYSCALL_REFILLBUFFER: Fájl olvasásánál, ha a megnyitott fájl beolvasott adatblokkja fel lett dolgozva, ezzel kérhető a következő adag BYTE betöltése. Ekkor az **A** regiszterbe az OPEN esetén megadott csatorna számát kell tölteni, majd a rendszerhívást el kell indítani. A végrehajtás után visszakapott paraméterek ugyanazok, mint a $23 / $24: SYSCALL_OPEN (_MEM) rendszerhívás esetén. Fájl írásakor, ha az adatpuffer betellett vagy már nem kerül bele több adat, akkor az **X** regiszterbe a pufferben található adatok kezdő pozíciója kerül, az **Y** regiszterbe az utolsó, még használt BYTE-é. Az **A** regiszterbe az OPEN esetén megadott csatorna száma jön, majd a rendszerhívás elindítható. A visszakapott paraméterek ugyanazok, mint a $23 / $24: SYSCALL_OPEN (_MEM) esetén.
- $28: SYSCALL_GETCHANNELPARAMS: Csatorna paramétereinek a lekérdezése. Az **A** regiszterbe a lekérdezni kívánt csatorna számát kell tölteni, majd a rendszerhívás elindítható. A visszakapott paraméterek ugyanazok, mint a $23 / $24: SYSCALL_OPEN (_MEM) esetén.
- $31: SYSCALL_CHANGEDISK: A meghajtó *firmware* „lemezcsere” funkciója, ezzel a lemezképek cserélhetőek egy külső fájlban szereplő lista alapján. Az **X** regiszterbe a lemezkép „sorszáma” kerül, utána a rendszerhívás elindítható. A visszakapott paraméter az **A** regiszterben $00, ha a csere sikerült. Ahhoz, hogy ezen funkció működjön, a meghajtó *firmware*-nek meg kell adni a cseréhez tartozó „csere-fájlt” (AUTOSWAP.LST). Ha a felhasználó az SD2IEC eszközön található gombokkal választott ki egy lemezképet, akkor ez a „csere-fájl” beállítás automatikusan megtörténik. Azonban a felhasználó közreműködése nélkül is megadható ez a „csere-fájl” az „XS:AUTOSWAP.LST” parancs segítségével. (Lásd a meghajtó *firmware* dokumentációját.)

7 VCPU-s programvégrehajtás hibakezelés

A VCPU-s programfutás több okból is befejeződhet, nem csak a fenti „SYSCALL_EXIT_xxx” rendszerhívásokkal. Amennyiben ismeretlen számú rendszerhívás történik, egy „97,VCPU ERROR,101,xx” hiba kerül beállításra, ahol az xx az ismeretlen rendszerhívás száma. Ha egyéb okból szakadt meg a programfutás, akkor egy „97,VCPU ERROR,100,xx” hiba lesz a végeredmény, ahol az xx értéke megegyezik a „ZC” paranccsal lekérdezhető VCPU állapot **INT** paraméterével. Ennek a BYTE-nak a bitjei különböző eseményeket jelentenek:

- B7: A meghajtó *firmware* fordítási hibája. Ezzel a programozónak / felhasználónak nem kellene találkoznia; nem megfelelő módon lett az eszköz firmware-je összeállítva.
- B6: Címhiba. Ha a VCPU az elérhető programmemórián kívüli címre ugrana, akkor ezzel a jelzéssel megszakad a programvégrehajtás.
- B5: Címhiba, érvénytelen címről történt rendszerhívás.
- B4: Érvénytelen utasításkód. Az „illegális” utasításkódok végrehajtása ezzel a jelzéssel megszakad.
- B3: Az eszközből eltávolították (vagy ha nem volt benne, behelyezték) az SD kártyát.
- B2: ATN aktív esetén a VCPU-s programvégrehajtás megszakad. Ez a viselkedés a megfelelő rendszerhívással ki / bekapcsolható!
- B1: Írás / olvasás címhiba. Ha a VCPU a használható memória / I/O / puffereken kívüli címről olvasna vagy oda írna, ezzel a jelzéssel megszakad a programvégrehajtás.
- B0: Rendszerhívás hatására a programfutás felfüggesztődött. Ez a jelzés a kilépés(ek) esetén fordulhat elő.

8 Megjegyzések

8.1 1 bit / 2 bit

A felhasználók egy része az SD2IEC meghajtót nem önálló eszközként használja, hanem a géphez van ezen felül csatlakoztatva még egy 15x1-es meghajtó is. A soros busz alapvető működése, hogy a számítógép ATN aktív jelére a perifériák a DAT vonalat alacsony szintre kapcsolják. Emiatt azon programok, amik az ATN-t is használják az adatátvitel alatt, általában csak akkor használhatóak, ha a soros buszon csak egyetlen periféria van! Ilyen esetre az SD2IEC meghajtók lehetőséget biztosítanak arra, hogy a soros buszról le lehessen őket „választani” az eszközök tényleges szétcsatlakoztatása nélkül is. Viszont fontos megjegyezni, hogy erre az 15x1-es (gyári) meghajtók esetén nincs mód! (A meghajtó egyszerű kikapcsolása nem elegendő, a soros buszról is le kell ilyenkor a perifériát csatlakoztatni!) Emiatt az SD2IEC-es adatátviteli rutinokat célszerű úgy implementálni, hogy csak különösen indokolt esetben használják az ATN-t! A fenti, *szinkron átvitel*ek közül az 1 bites verziók nem használják az ATN-t, preferált kommunikációs módnak a csak a CLK / DAT vonalakat használó megoldások tekintendők!

8.2 Pufferek / memória

A VCPU programmemóriája a meghajtó *firmware* adatpuffereinek a területe. Ezért oda kell figyelni arra, hogy a fájlkezelés alkalmával használt pufferekbe ne kerüljön programkód, mert az felül fog íródni. A programozónak (jelenleg) nincs ráhatása arra, hogy milyen művelet melyik puffereken keresztül fog végrehajtódni, viszont – szerencsére – a pufferfoglalás / felszabadítás kiszámítható módon megy végbe. A meghajtó *firmware* az elejétől, sorban egymás után foglalja a puffereket, viszont van pár dolog, amire oda kell figyelni. Ha a felhasználó (vagy maga a program) lemezképpel dolgozik, akkor ennek a kezeléséhez a meghajtó lefoglal egy puffert „belső” használatra. Ez általános felhasználásnál a 0. számú puffer lesz. Viszont ez a puffer akkor is foglalt marad, ha a felhasználó „kilépett” abból a lemezképből, mint könyvtárból! Ezen felül egy megnyitott fájl szintén lefoglal egy puffert, ez ebben az esetben az 1. számú puffer lesz. De ha ez a fájl nem lemezképen belül van, és a felhasználó eddig még nem is használt lemezképet, akkor ez a puffer a 0. számú lesz! Emiatt ilyen esetben a majd használt puffer számát dinamikusan célszerű kezelni.

Ez a két puffer a `$0000..$00FF`, illetve `$0100..$01FF` VCPU-s memóriatartomány lesz. Emiatt szükségszerű, hogy a nulláslap, illetve a verem áthelyezhető legyen, erre a VCPU ad is

lehetőséget. A fennmaradó szabad memória a kevés pufferrel rendelkező meghajtók esetén elég szűkös, 1 KBYTE, ez az 1541 memóriájának a fele. Viszont a lemezkezeléssel kapcsolatos feladatokat megoldja a meghajtó *firmware*, a bonyolult (és ezáltal sok memóriát igénylő) programkódok nagy része itt szükségtelen.

A pufferek számát az SD2IEC meghajtóban levő mikrovezérlő RAM-jának a mérete határozza meg. Mivel ennek a mérete eddig nem igazán számított, ezért a felhasználók többségének olyan SD2IEC meghajtója van, amiben a benne levő mikrovezérlő csak 6 puffert tesz lehetővé. Emiatt a programozónak célszerű ennyivel számolni. Ha szükség van több memóriára, a jelenlegi felhasználók jelentős részének nem lesz hozzá megfelelő hardvere, így az adott programot nem fogják tudni futtatni.

8.3 Lemezképek / könyvtárak

A különböző lemezképeket a meghajtó *firmware* segítségével egyszerű könyvtárként lehet elérni. Könyvtárat váltani a „`CD:...`” paranccsal lehet. (Lásd a meghajtó *firmware* dokumentációját a részletekhez.) Ezen felül a könyvtárak / lemezképek közötti váltás működik az „`AUTOSWAP.LST`” (vagy egyéb, külön megadott) fájl alapján is. (Normál működés közben a készülék nyomógombjainak a segítségével lehet a megadott listában szereplő lemezképek / könyvtárak között lépkedni, a VCPU-s programfutás alatt a `SYSCALL_CHANGEDISK` (`$31`) funkciót hívva érhető el ugyanez.) Fontos: **könyvtár / lemezkép váltás esetén az összes nyitott fájl / kommunikációs csatorna lezárásra kerül!** Ha egy lemezkép szektorszintű kezeléséhez nyitva volt egy kommunikációs csatorna, azt a csere után újra meg kell nyitni!

8.4 Háttértár sebesség

Ugyan a töltési (mentési) idő jelentős részét a meghajtó ↔ számítógép közötti kommunikáció teszi ki, de nem elhanyagolható az SD kártya, illetve a rajta levő fájlrendszer kezelésére fordítandó idő sem. Viszont ezek az idők függhetnek a kártya típusától, a rajta levő fájlrendszer paramétereitől, a fájl elhelyezkedésétől, stb. A programozónak nem szabad arra számítania, hogy az általa tapasztalt sebességet az összes felhasználó konfigurációja is el fogja érni!

8.5 VCPU bővített / „mikró” utasítások kódjai

Az elterjedtebb SD2IEC meghajtók hardverének szűkös erőforrásai miatt a bővítésnek nem célja egy már meglévő, régi CBM meghajtóval való kompatibilitás. Emiatt a CPU emulációnak az

eredeti 6502-vel való 100%-os egyezősége sem szükségszerű. A „nem dokumentált” utasításkódok megvalósítása is ebből az okból hiányzik, de ezen utasítások használhatósága egyébként is korlátozott. Mivel nem kell egy, már létező hardverrel kompatibilisnek lenni, emiatt érdemes lenne a „nem dokumentált” utasítások helyett a 65**C**02 CPU plusz utasításait / címzésmódjait implementálni. A VCPU extra utasításainak a kódjai úgy lettek kiválogatva, hogy azok a 65C02 CPU nem használt utasításkódjai helyére kerültek, emiatt a későbbiekben ezen CPU új utasításai / címzésmódjai is megvalósíthatóak! (A kisebb programmemóriával rendelkező meghajtók esetén erre jelenleg nincs szabad hely.)

9 A minta forráskódokról

9.1 A „#”

A VCPU az eredeti 6502-s utasításkódokon kívül ismer néhány új utasítást is. Ezek között a paraméterrel rendelkező utasítások jellemzően „bennfoglalt” (*immediate*) címzésmódot használnak. Ezt a címzésmódot az eredeti 6502 *assembler* szintaktika „#” jellel jelöli (pl. `LDA #$55`), a dokumentáció előző részeiben is ilyen módon vannak ezen utasítások bemutatva. Jelenleg nincs olyan *assembler*, ami a VCPU bővített utasításait ismeri, ezért a használatukhoz készült egy makró-definíciókat tartalmazó forrás-fájl, ami segíti a programozást. Viszont az *assembler*ek makró-definíciónál jellemzően nem tudnak olyan paramétert feldolgozni, ami *szám* (vagy azt reprezentáló címke), és „#” jellel kezdődik! Emiatt a mintaprogramokban az ilyen, makróként definiált utasítások paraméterei elől a „#” jel el van hagyva. Tehát

`BREAK #$00` helyett `BREAK $00`

alakban van az összes ilyen utasítás leírva. Viszont ez nem jelenti azt, hogy ezen utasítások paramétere (nulláslapos) cím lenne! Mivel az összes „új” utasítás csak egy fajta címzésmóddal használható, ezért ez nem okozhat félreértést. Ha a későbbiekben lesz olyan *assembler*, ami ezen utasításokat direkt támogatja, ott az eredeti szintaktika szerint a „#”-et használó írásmódot célszerű megvalósítani, de kompatibilitási okból a „#” nélküli írásmód is elfogadható.

9.2 A mintaprogramok támogatott architektúrái

A mintaprogramok négy fajta architektúrára fordíthatóak, ezek a VIC20, a C64, a C16 / C116 / plus/4 (C264 széria), illetve a C128. A gépek KERNAL-on keresztüli meghajtó-programozása megegyezik, de a minták között sok olyan program van, ami a meghajtó↔számítógép közötti adatátvitelt mutatja be, ezek viszont hardverfüggőek. A mintaprogramok célja inkább az SD2IEC +

VCPU, mint a számítógép oldali programozás bemutatása, ezért a VCPU oldali programkódok – az egyszerűség jegyében – mindegyik platform esetén megegyeznek. Viszont ez azt is jelenti, hogy az egy-egy adatátvitelt bemutató minta nem biztos, hogy az összes platformon optimális megoldás! (Pl. a számítógép oldali hardver platformonként más-más bitsorrendet kívánna.) A programozási minták arra a célra (is) készültek, hogy segítsék a programozót a választott platformhoz leginkább optimális adatátvitel implementálásában.

9.3 A mintaprogramok licence

A mintaprogramok azért készültek, hogy esetleg hasznosak lehetnek. A **„semmire sincs garancia, még az alapvető működésre sem”** megjegyzés ismeretében szabadon, részeiben vagy egészében, mindenféle korlátozás nélkül felhasználhatóak.

9.4 A mintaprogramok leírása

A mintaprogramok a VCPU bővítés funkcióinak a tesztelésére készültek, de esetleg hasznosak lehetnek az eszköz programozásával kapcsolatos kérdések megválaszolásában. A programok mindegyike először a géphez kapcsolt perifériákat nézi végig, megszámolva a soros buszra kapcsolt eszközöket. Kikeresi az SD2IEC meghajtót (több találat esetén az utolsót), majd a programhoz tartozó feladatokat ezzel a meghajtóval hajtja végre. Ezen programok a következő funkciókkal rendelkeznek:

- `A-DETECT`: A megtalált SD2IEC meghajtónak kiírja a „hosszú” verzióját („`X?`”). Ellenőrzi a VCPU támogatást („`ZI`”), a kapott értékeket megjeleníti. Majd kiírja a pufferek állapotát („`ZB`”), illetve a VCPU aktuális állapotadatait („`ZC`”).
- `B-SHOWMEM`: Az SD2IEC meghajtó VCPU által kezelt memóriáját jeleníti meg („`ZR`”).
- `C-PRINTIO`: A VCPU által látott I/O területet jeleníti meg, ehhez (és a többi mintához) már a meghajtóba letöltött megfelelő program futtatására van szükség.
- `D-BUTTLED`: Az SD2IEC két LED-je („BUSY” / „DIRTY”) vezérelhető a két nyomógomb („PREV” / „NEXT”) segítségével. Illetve a számítógép CLK / DAT vezetéke is a LED-eket kapcsolgatja (**C** + **V**, illetve **D** + **F** billentyűk). Kilépéskor a meghajtó státuszát lekérdezi, ez az ATN aktiválásával jár, a VCPU erre szakítja meg a programfutást. Emiatt a lekérdezett státusz a megfelelő hibát adja vissza. Ezek után a VCPU állapota is megjelenik.
- `E-SR1B-PIO`: A gép az SD2IEC felé elküld 256 BYTE-ot, majd ezt az adatcsomagot a meghajtó visszaküldi a gépnek. Az leellenőrzi, hogy ugyanazt kapta-e vissza, mint amit küldött. Ha igen, kezdődik a küldés elölről. Ez a teszt addig fut, ameddig a felhasználó ki nem lép, vagy az összehasonlítás során eltérés nem keletkezik. A képernyőn a meghajtótól visszakapott adathalmaz megjelenik. Hiba esetén a hibás BYTE színe megváltozik. Ez a teszt az adatátvitel során csak a CLK / DAT vonalakat használja, a DAT-on az adatbitek mozognak, a CLK segítségével a számítógép ütemezi az adatátvitelt. A VCPU oldalán a kommunikáció az I/O regiszterek segítségével, „hagyományos” módon van megvalósítva. Ez az adatátviteli mód nem ajánlott!

- F-SR1B-USR: Adatátvitel teszt, a mozgatott adatok megegyeznek az E-SR1B-PIO tesztnél leírtakkal. Ez is csak a CLK / DAT vonalakat használja. A VCPU oldalán a kommunikáció az USND1 / URCV1 utasításokkal van megvalósítva. Ajánlott adatátviteli mód!
- G-SR1B-URE: Adatátvitel teszt, a mozgatott adatok megegyeznek az E-SR1B-PIO tesztnél leírtakkal. Csak a CLK / DAT vonalakat használja. A VCPU oldalán a kommunikáció az UDTTA / UATDT utasításokkal van megvalósítva. Ajánlott adatátviteli mód! A DAT vonalon a bitek sorrendje fordított az előző adatátvitelekhez képest.
- H-RECVTIME1B: Adatfogadási idő teszt, meghatározott számú BYTE fogadása és ellenőrzése minden egyes képmegjelenítés alatt. Csak a CLK / DAT vonalakat használja.
- I-SR2B-PIO: Adatátvitel teszt, az E-SR1B-PIO két bites megfelelője. Egy lépésben a CLK / DAT vonalon 2 bit mozog, az ATN segítségével ütemezi a gép az adatátvitelt. Az ATN-t használó két bites átviteleknél csak egy meghajtó (itt az SD2IEC) csatlakozhat a számítógéphez a soros buszon! A VCPU oldalán a kommunikáció az I/O regiszterekkel történik, ez az adatátviteli mód nem ajánlott!
- J-SR2B-USR: Adatátvitel teszt, az F-SR1B-USR két bites megfelelője. Az adatmozgatás módja megegyezik az I-SR2B-PIO-nál leírtakkal. A VCPU oldalán a kommunikáció az USND2 / URCV2 utasításokkal van megvalósítva. Ajánlott adatátviteli mód!
- K-SR2B-URE: Adatátvitel teszt, a G-SR1B-URE két bites megfelelője. A VCPU oldalán a kommunikáció az UCDTA / UATCD utasításokkal van megvalósítva. Ajánlott adatátviteli mód! Az adatátvitel alatt a bitek sorrendje fordított az előző adatátvitelekhez képest.
- L-RECVTIME2B: Adatfogadási idő teszt, a H-RECVTIME1B 2 bites megfelelője.
- M-LOADTST1B: Fájltöltés teszt. 1 bites adatátvitel a CLK / DAT vonalak használatával. A meghajtóban szükséges egy SD kártya, az aktuális könyvtárban a „TSTDAT2M.SEQ” tesztfájllal. Ez egy 2 MBYTE-os, véletlen-számokból álló fájl. A teszt ezt a fájlt olvassa végig négyszer. Az olvasások között a blokkméret (254 vagy 256 BYTE), illetve az ellenőrző összeg számítása (számolja vagy nem) a különbség. A folyamat alatt megméri azt az „időt”, ameddig az olvasás tart. (Ez az „idő” a használt architektúra megszakításainak a száma, ami VIC20 / C64 esetén ~60 db, (PAL) C16 / C116 / plus/4, illetve (PAL) C128 esetén ~50 db másodpercenként. A számolt értéket ennyivel osztva megkapható a töltési idő másodpercben.) A kapott értékek közvetlenül nem alkalmasak a platformok közötti összehasonlításra!
- N-LOADTST2B: Fájltöltés teszt. Az M-LOADTST1B 2 bites megfelelője, a CLK / DAT és az ATN vonalak használatával.
- O-DISKIMGS: Lemezkép-kezelés teszt. A meghajtóban szükséges egy SD kártya, az aktuális könyvtárban a „VCPUTSTDSK1.D64” illetve „VCPUTSTDSK2.D64” lemezkép-fájlokkal. A teszt a „CD:...” parancs segítségével belép először az első, majd a második lemezképbe, majd a bennük található 3-3 fájlt végigolvassa direkt szektor-olvasás parancsokkal. A beolvasott fájlok tartalmából ellenőrző összeget számít, amit (egyéb adatokkal együtt) a futás végén visszaad a számítógépnek, ami megjeleníti / ellenőrzi azokat.
- P-AUTOSWAP: Lemezkép-kezelés teszt. Az O-DISKIMGS tesztnél olvasható feladatokat hajtja végre, azonban a lemezkép-cserét az „AUTOSWAP.LST” fájl alapján végzi (ez a fájl is szükséges az SD kártya aktuális könyvtárába a lemezképek mellé).
- Q-BENCHMARK: Fájlolvasás sebességteszt. A VCPU-s tesztprogram a meghajtó *firmware*-rel végigolvastatja a töltéstesztek alatt használt „TSTDAT2M.SEQ” tesztfájlt, a számítógép

közben megméri az olvasási „időt”. Mindezt 254 és 256 BYTE-os blokkmérettel is végrehajtja. A teszttel a meghajtó „nyers” fájlolvasási sebessége mérhető! A kapott „idő” a fájltöltés-teszteknél ismertetett módon számítódik.

- `Y-LINEDIAG`: Az SD2IEC oldaláról a soros vonalak tesztje. Diagnosztikai program, a későbbiekben lehetséges az eltávolítása!
- `Z-VCPUCTST`: A VCPU mag tesztje, az emulált 6502-s utasítások futásának az ellenőrzésére. Nem „mindenre kiterjedő”, teljes teszt! Jelenleg az összes fajta (eredeti 6502-s és bővített) utasítás legalább egy fajta címzésmóddal fut / ellenőrizve van, illetve az összes fajta címzésmód legalább egy utasításban tesztelve van. (A „mikró” utasítások itt nincsenek tesztelve, azok nagy része az adatátviteli tesztek alatt van használva / ellenőrizve.) Az esetleges VCPU utasítás hibák a későbbiekben külön tesztet kaphatnak ebben a programban a javításukkal egy időben.

10 Ismert hibák

- A VCPU-s programvégrehajtás csak a RAM területről megengedett! De a végrehajtandó utasítás címe csak akkor van ellenőrizve, ha feltétel nélküli ugrás hajtódik végre. Emiatt egyszerű programfutás alatt a **`PC`** kifuthat a RAM-nak kijelölt területről. Illetve a feltételes elágazó utasításokkal is ki lehet ugrani az engedélyezett tartományból. Ilyen esetben „nem definiált” a működés. Az esetleges érvénytelen memóriacímre történő írás nem hajtódik végre (ahogy normál programfutás közben sem), ezért a meghajtó *firmware* normál működését egy ilyen eset nem befolyásolja. Az ilyen, „hibás” memóriacímen levő rendszerhívás utasítás (`$00`: `BREAK` kód) szintén nem hajtódik végre.
- A hagyományos megszakításkezelés hiányzik! Viszont van több olyan esemény is, amikor a VCPU-s programvégrehajtás megszakad. (Ilyen esemény például az SD kártya eltávolítása.) Ezen események vizsgálata nem történik meg minden utasítás végrehajtásakor, csak olyan utasítások után, amiknek a segítségével „programhurok” készíthető. (Ezek jellemzően az ugró utasítások / várakozó „mikró” utasítások.) Ezért az ilyen eseményeknél a program végrehajtása nem akkor szakad meg, amikor az esemény történt, hanem az azt követő valamelyik ugró utasításnál! Ilyenkor az utólag lekérdezett VCPU állapotban a **`PC`** értéke alapján nem derül ki, hogy melyik utasítás futásakor történt az adott esemény.
- Az I/O területet elérő utasítások futásideje függ a választott regiszter címétől. Ugyanazon regiszter elérése mindig ugyanannyi időt igényel, a kiolvasott / beírt tartalom nem változtatja meg a futásidőt.
- A `BTASC` utasítás egyszerű ciklusokkal számolja át a bináris értéket ASCII karakterekké, emiatt a futásidő (ezen utasításnál egyedüliként) függ az átalakítandó értéktől! Sebességkritikus részen emiatt ezen utasítás használata kerülendő.
- Az eredeti 6502-s utasítások futásideje (kevés kivételtől eltekintve) nem definiált. Esetleges optimalizálások miatt a későbbiekben ezek változhatnak!

Tartalomjegyzék

Változáslista:
211108: „PERV” → „PREV”
211112: `ULBIT` utasítás
220407: „`ZE`” parancs indulási idő kiegészítés