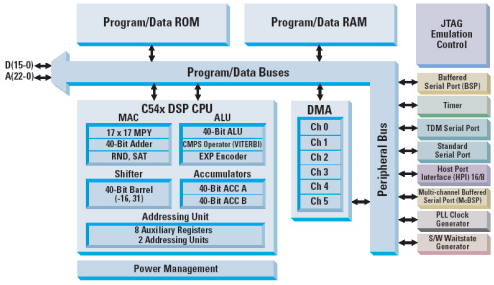
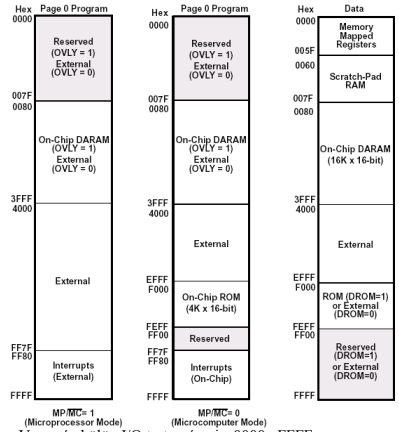
TMS320C54x™ DSP Generation, Fixed Point

Készítette: Nagy István buenos@freemail.hu



C54x™ DSP Generation Block Diagram

A memóriatérkép:



●Van még külön I/O tartomány is. 0000...FFFF.

●A memóriák 128 híszavas lapokra vannak osztva. Datamem: Az ST1.DP regiszterrész adja meg az aktuális lap sorszámát.(max 512db)

●hozzáférés-vezérlése: Az on-chip daram hova tartozzon: pmst.ovly - bit alapján (0-csakadatmem,ekkor a pmem területen külső terület van. 1-adat és prog címterületen is), az on-chip rom hova tartozzon: pmst.drom (0: csak p.m., 1:prog.m. és d.m. területhez is). A programjaink fussanak a daramból, vagy külső ramból.

●**stack**: SP mutat rá, inicializálni kell. töltődéskor csökken a cím. Külön szekció kell neki (.usect) az adatmemóriában

●MP/MC-láb:uP/mikrocomputer mód. (MC:külső a progmem, nincs boot sem)

●Reset vektor: FF80h Az on-chip rom-ban van 1 bootloader program.Ez az F800-tól kezdődik (MC mód). IT vektortábla: FF80h...FFFFh (ez programból módosítható. Célszerű is MC módban, startup-kor, mert az it táblánkat nem tudjuk a rom-ba tenni, csak ramba (daram vagy ext.)). FF80-ról F800-ra ugrik:bootloader indítl.

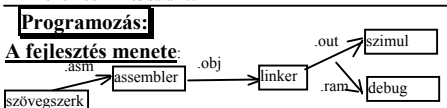
●Az on chip rom FF00-tól FFFF-ig tart, tehát a resetvektor is ebben van. ott van egy ugrás (MC mód)A bootloaderre. Találhatók még benne adattáblázatok is, pl szinusz(FE00...FEFF 256db minta), µ/A-Law tábla.

●Kiterjesztett programmemória: Nem csak 64k érhető el, hanem 8MB, 128db 64kB méretű lapokra osztva. Normál használat: csak a 0.lapot használjuk. 23címvonal megy ki az ic-ből. A PC kiegészítésére van az XPC regiszter (=lapsorszám,=felső 7bit).Külön utasításokat kell használni ugrásnál: FB, FCALL, FRET... (F=far=távoli ugrások) Ha az ovly=1, akkor minden lapnak a felső fele ugyanaz. ON-chip rom csak a 0.lapon van.

●Az 5402-ben nincs saram!!

Programozás:

A fejlesztés menete:



Memória-allokáció, direktívák:

.text	Programkód szekció kezdet jelzése. Az ilyen szekciókat a linker sorban egymás után helyezi el a programmemóriában. (rom, vagy ram a ramba a bootloader teszi.)
.bss x,n,b	az x nevű változónak foglal n darab helyet. Adatmemóriában. A linker .bss nevű szekcióba teszi az adatokat. A b elmaradhat. Jelentése: Ha b>0, (b=1)akkor laphatáron kezdődik a változónk allokálása, és 1befüggő a területe.
.data	Ez után konstansokat adhatunk meg. (név .set érték) Ezeket a programmemóriában helyezi el (rom vagy ram).
név: .int adat1,adat2... **név: .word adat1,adat2...**	16 bites konstansok megadása. A megadott névhez-hez a linker által rendelt címtől kezdve sorban helyezi el a megadott adatokat. (szám, vagy 'k'-karakter, "szöveg") név=első címe.Ha alá írunk még egy .word-ös sort, név nélkül, az az előző után kerül a pmem-ba.
név: .string "szöveg"	szöveg elhelyezés pmem-ba. cimk=név.
név .set érték **név .equ érték**	csak a fordítónak szól. név=szám. De változóinkat is elnevezhetjük így, csak a linker ne pakoljon ezekre a helyekre!
.sect szekciónév	Névvel ellátott bármilyen szekció kezdete. Linkelésnél erre a névre hivatkozunk Lehet prog., vagy adatmemóriába is. konstansoknak, kódnak
címke: .usect "szekcnév",n	n darab változónak foglal helyet a "szekcnév" nevű szekcióban. Az elsőre a címke mutat (változónév). Adatmemória.
.space n*16	beszúr n db szót, 0-kkal kitöltve. (pmem)

●Szám megadás: #1234 -decimális, #100101b -bin., #0a5abh -hexa

●A reset/it vektorokat tegyük külön szekcióba!

●A stack-nek is kell helyfoglalás! (.usect)

●A program vége: .end

●**Lapozás kezelése**: Hogy tudjuk, hogy melyik változónk, melyik memórialapra kerül, nevesített (.usect) szekciókba tegyük őket. Ha az összes bss <lapméret (128szó, max 512lapos a címtartomány. DP állítja az aktuális lapot), akkor a 0. lapon lesz mind. Csak ekkor nincs probléma vele.

Lapváltás oda, ahol a változónk van: LD #változó,DP Ekkor a változó felső része kerül a DP-be. (De ha ld #16bites_szam,ld : akkor már a szám alja kerül bele.) Utána a LD változó,A (vagy @változó) már helyesen hivatkozik a változóra. Több mint 128 változó esetén alkalmazzunk pointert, vagy abszolút címzést! (külön ilyen utasítások vannak. pl MVDK, MVMM... vannak utasítások, amik csak lapon belül címeznek. Még akkor is, ha cimkével hivatkozunk változóra.)

●**Többfájlos project**be kell a forrás tetejére .def változó1, változó2...(itt megadott változó, cimke, de máshol is látni akarjuk), és .ref változó3, változó4...(a máshol megadott változók láthatóvá tétele az aktuális forrásban.): Vagy mindkettőbe: .global valt1,valt2...

Linkelés:

Elhelyezzük a memória szekciókat a memóriákban. Megadjuk: a memóriadarabokat, kezdőcímeiket, méretüket, (MEMORY {...})szekciókat, és hogy melyik szekció melyikbe kerüljön. (SECTIONS {...}) Ez a linker comman fájlban lesz. (*.cmd) Minta:

```
MEMORY
{
    PAGE 0:
        VECT:   origin = 0FF80h, length = 0080h
        PROG:   origin = 0800h, length = 1000h
    PAGE 1:
        DARAM :  origin = 0080h, length = 2800h
}
SECTIONS
{
        .ugrtab:  {} > VECT  PAGE 0
        .text :   {} > PROG  PAGE 0
        .bss  :   {} > DARAM PAGE 1
}
```

Címzés: (utasítások operandusai).

●**azonnali**: #szám -cím megadása. 16bites: full 64k, 8 bites adattal:lapon belüli cím. Ha címkét használunk, akkor is igaz. Smem-nek jelölik a könyvben az operandusokat.

●**abszolút**: az operandusokat dmad, pmad -nak jelölik a könyvben. Lapfüggetlen címzés. szám, vagy címke.

●**akku**: pmem akku általi címzése.

●**indirekt**: ARn segédregiszter általi címzés. (Teljes 64k) pl. *AR5. Lehet auto increment (*AR5+, *AR5+0 ezutóbbi: AR0-tartalmával növeli), decrement, cirkuláris címzési (*AR5+% vagy *AR5+0% ezeknél a BK regiszter a bufferméret) opcióval. Ezek memóriahozzáférés után változtatják ARn tartalmát. A könyvben xmem, ymem -nek jelölik az operandusokat. Báziscím használata: pl. *+ARn(bázisszám)%. Csak ARn módosítása: MAR ARn%+...

●**direkt**: DP (cím=DP|megadott cím) vagy SP (SP+megadott cím) segítségével. pl lapon belüli címzés. A ST1.CPL -bit től függően: 0:DP-relatív, 1:SP-relatív. A relativitást jelzi a:@-jel. Elmaradhat. Ez a legegyszerűbb. Több mint 128 változó esetén alkalmazzunk pointert, vagy abszolút címzést!

●**verem**: push, pop -stackműveletek. (PSHD, POPD)

●**mmr**: Az mmr-eket (memóriába ágyazott spec. regiszterek) közvetlenül a nevükkel el lehet érni, ha a progi elején volt: .mmregs.

●**változók kezelése**: #név: a változó teljes címe. A névre hivatkozás (.bss változó): lapon belüli címzés. De ha pl x .set cím akkor full 64k a címzés. Ekkor a dolgokat nekünk kell allokálni a memóriába, vigyázva, hogy a linker ne tegyen ugyanoda semmit sem.

Utasításkészlet:

1.) Adatmozgatás, értékadás:

Utasítás x,y x->y	hova				
	mmr	**akku**	**Dmem**	**Pmem**	**i/o**
honnan **mmr**	mvmm	ldm	mvmd		
honnan **akku**	stlm	ld	st		
honnan **Dmem**	mvdm,ld	ld,mvdm	mvdd,mvdk,mvkd	writa,mvdp	portw
honnan **Pmem**			reada,mvpd		
honnan **i/o**			portr		
honnan **konstans**	stm	ld	st		

●**Utasításfüggőek** a címzési lehetőségek.(fullcím, laponb.cím,indirekt). Indirekt (mutatós)címzés mindegyiknél van, de valahol csak az van. pl.: ld,st csak laponbelüli, vagy indirekt, mvdd (ind->ind), mvdk (laponb.->lapf.len), mvkd (lapf.->laponb.), mvdm(lapf.len->mmr), mvmd(mmr->lapf.len), portw (laponb.memcím->full i/o-portcím), reada(akkuban tárolt pmem-cím->laponbcímű.dmem)

●Egyes utasítások nem tudnak minden mmr-be/ből . (pl. az ld,st)

●**A shift**: negatív szám: osztjuk, +:szorozzuk a változót 2^n-nel.

●Vannak **duplaszavas** (dld,dst), és kettős operandusú, párhuzamos műveletek (pl. ld...||st...).

●Stl, sth az akkuk alsó, vagy felső részével dolgoznak.

●Meg lehet adni 1es helyeken shift paramétert. (pl. ld a,8,b vagy a paraméter lehet a TS regiszter tartalma)

●**Címzések**: Xmem,Ymem (*ARn mutatóval), Smem (lapon belüli, cím vagy cimke), Dmad, Pmad(lapfüggetlen), src (A,B akkumulátorok),lk (16bites konstanssal).

●**Ez konstans**: #x -ez az x címét teszi be konstansként adatnak.

●**Változatok**: stl,sth... akku felső/alsó felét írja ki

●**mmr-nél**: Írás után olvasás: pipeline ütközés lehet, ezért az írás legyen **védett írást** használjunk! Azoknál nincs ilyen probléma. pl mvmm, mvdk... pl. **AR regiszter használat előtti feltöltése jóval előbb legyen!!! Ha máshogy nem, NOP-okat tegyünk be!** Védett írás is csúszhat, ha előtte sima írás van. tehát nem kell mindíg védett.

Előfordulhat, hogy egyik utasítás írná, a másik Ivasná ugyanazt, és akkor nem íródik be az új érték a regiszterbe. (pl ld k,ARn...mvdk)

●**törtek megadása**: 0,xxx megadása: név .word 32768*xxx/1000

●A használt portjainknak a .EQU-val adjunk nevet!

●**Nullázás**: LD #0,A... ilyesmi

2.) Aritmetikai műveletek:

●**összehasonlítás**: CMPR módkód,ARn ARn-et AR0-val hasonlítja. Ha a feltétel igaz, TC=1 lesz. (módkód:00...11:=,<,>,!=). CMPM változó,#szám: Ha a laponb.címzett változó=szám,TC=1 lesz.

●**összeadás, kivonás**: ADD változó,akku vagy ADD szám,akku. lehet shiftelni közben. akku=akku+változó. meg lehet adni +paraméternek:cél akku. (pl. add x,a,b) egyesek csak elöjeltelen számokkal műxenek. Átvitel figyelembevétele: ADDC, és SUBB.

●**max, min**: MAX cél : cél akksi =max(a,b). többől max/min: új adatokat az a-ba, eredményeket a b-be. (max cél)

● **abszolút érték** :ABS akksi,akksi.

●**szorzás**: MPY. eredmény az egyik akksiba megy: +adjuk. változó*T, vált1*vált2, vált*konst, konst*T. Ahol 1 változó van: laponb.cím, ahol 2: ott 2 pointer kell. (mpy x,y,a) Normális előjelkezeléshez az st1.frct bitet állítsuk előbb 0-ba!

●**osztás:** nincs. de van feltételes kivonás: SUBC. Ebből kell. Ez előjeletlen, előtte flagekkel kezeltetni kell az előjeleket.

●**MAC**: akksi+változó*T, vagy akksi+vált1*vált2, akksi+vált*konst, akksi+konst*T. Az eredmény az (valamelyik) akksiban tárolódik. (MAC_x,y,a)

Túlcsordulás kezelése: Ha ST1.OVM=1 akkor 32bit túlcsordulás esetén betelít az akksi. MAC-nál nem jó ha betelít (szaturál), mert az akksi guard bitjeit is kihasználhatjuk. +oldás: végső akkumentés előtt szaturáljunk csak: SAT akksinév. Szaturálás nélkül 1 túlcsordult jal mentése hibát okoz. Legjobb a non-gain-rendszer.

●**logikai**: AND szám, akku, vagy AND szám, akku vagy AND akk1,akk2 -és kapcsolat.Lapon belül címez. Ha nem adatmemóriából szedi az operandust, lehet benne shift. Eredm.az akksiba megy!Ugyanígy működik az: OR, és a XOR.

ORM #szám, dmem.laponb.cím, ANDM: Változó és szám logikai művelete, akksi nélkül.(Eredm vissza a változóba)

●**négyzet**: SQUR: (változó)^2=akksi (squr x,a)

3.) Ugró utasítások:

●**sima ugrás**: B cím vagy BC cím,feltételkód. BD: késleltetett ugrás.

aeq	A=0
beq	B=0
aneq	A!=0
aleq	A<=0
agt	A>=0
bio	bio-láb=0
c	carry=0
ntc	tc=0

●**kisciklus**:RC regiszter tartalma alapján(annyiszor) ismétel 1 utasítást. (RPT) A ciklusszámot megadjuk: vagy szám, vagy egy változó értéke. (a változót lapon belül címzi) Nem kell állítgatni az rc-t. Max 65535.

RPT hányszor utasítás

●**közepes ciklus**:(rptb utasítás) A BRC-regiszter tartalma a ciklusszám. Ezt be kell állítani előbb: stm #szám,brc A pipeline miatt a címkét tegyük egyel lejjebb, és az rptb után nem cimke, hanem cimke-1 -re megy az ugrás-2szavas utasításoknál. Max 65535

RPTB cimke
ut1
ut2
cimke: ut10

●**nagyciklus**: BANZ-utasítás, valamelyik AR regiszter tartalma a ciklusszám. Ez ott leszámol.0-kilép.Előtte bekell állítani az AR-t!

cimke: ut1
ut2
...
banz cimke, *ARn-

●**szubrutinhívás**: CALL cimke. feltételesen: CC cimke,feltételkód. (mint a B utasításnál) Visszatétés: RET, vagy RETD. RETE ugyanez, csak engedélyezi az it-ket.

●**szoftver interrupt**: INTR sorszám. Cím=báziscím+sorszám*4. B.c. defaultból =FF80. Vagy: TRAP sorszám. A visszatérés ugyanaz(ret...). Vannak külön szoftver it vektorcímek (sorszám=2...15), de lehet más it-vektorra is ugrani szoftverből, sőt resetre is lehet (reset vektor).

●**feltételes utasításvégrehajtás**: XC darab, feltétel1,feltétel2... A következő néhány darab utasítás(szó)t végrehajtja, ha a feltétel igaz.A feltételnek min.2vel korábban meg kell lennie!

4.) késleltetett utasítások: A sima Utasítás mnemonikjának a végére D-t írnak. (pl. BD,) A késleltetett BD-t 2 utasításszóval előbbre kell tenni, mint ahonnan ugorni akarunk. Az ugrás után kiürülne a pipeline, ezért az üres hely kitöltésre van a 2 végrehajtódó utasítás a BD után.

5.) kettős operandusú: Egy utasításban adjuk+ egy művelet mindkét operandusát. MAC, és MPY utasításoknál. Címzé csak AR2...AR5 -ön keresztül lehet. Jó gyors. pl mac *ar2+,*ar3+,a

6.) duplaszavas: 32bites adatokkal dolgoznak. A sima Utasítás mnemonikjának az elejére D-t írnak. pl DADD, DLD, DST... Az adatok 2. szava az első után kell hogy legyen, ha az első páros címen volt. Nem lassab a végrehajtás. (HW trükk)

7.) párhuzamos utasításpárok:pl. LD ...||ST..., de van sok más is.

8.) bitkezelés: BIT cím,kód: (pl bit *ar2,11)Az ST0-nak a tc regiszterébe másolja a címen lévő adat megadott bitjét. Csak indirekten lehet címezni. A kód: a bit sorszáma, DE visszafelé!!! 0000=15.bit. A tc felhasználható feltételként ugráshoz. A BITT cím esetén a címzés lapon belüli, a kódot a T-be kell előbb tenni.

●SSBX bitnév (1-be,tilt), RSBX bitnév (0-ba,engedélyez): Csak az ST0, ST1 bitjeit egyenként tudják állítani. Más regiszteréknél fondorlatosabban kell eljárni: 1-be állítás: ORM #0000010000000000,mmrnév, (ott 1, ahol a bit van) 0-ba: ANDM #1111111011111111,mmrnév (ott 0, ahol a bit van). nem mmr-eknél OR-t kell használni, majd az akksiból az eredményt visszaírni. LSB=0.sorszámú bit az adatlapon.

●Egyes státuszrészeket, pl st1.asm (ASM-néven), (csak st0, st1 részeit) simán lehet regiszterként írni, mégha azok nem 16 bitek is. (másokat nem) De ekkor nem ugyanaz vonatkozik rájuk, mint más mmr-ekre. pl ld utasítással kezelni. ld #szám,dp. csak őket lehet így kezelni.

9.) Egyéb utasítások:

●SFTA honnan, mennyivel, hova -shiftel, egyik akkuból a másikba.

●Kerekítés: R-toldalék. A 32 bites eredmény felsejét kerekíti, az alja alapján. pl. MACR

●Forgatás: ROL akksi, ROR akksi. Balra/jobbra, carry-vel.

●Buffer: Lineáris buf. öregbítés: DELAY *ARn- . Ezt a bufferméretszer ismételve, előtte a buffer elejére állítjuk ARn-et.

Cirkobuffernek csak a címzést kell öregbíteni, és külön szekció kell neki. Kell 1 író, és egy olvasó pointer. mindkettőt *ARn%+ nel címezzük. Ha = a 2 pointer: buffer full, vagy buffer empty

●Szimmetrikus **FIR-szűrő** számítása: (rendszeregyenlet kiszámítása fc-nként egy mac-sorozat. Ciklusba.) Spec utasítás van rá: FIRS_Xa,Xb,együtthatótábla. Xa, Xb-t indirekt, cirkuláris címzéssel használjuk (*ARn+%).A bejövő adatokat 2 bufferben tároljuk: Xa:X[n...n-N/2], és Xb: X[n-N ...n-N/2-1]. Tehát a minták újabbik fele Xa, a régebbik Xb. Y régi értékeit nem kell tárolni. Mivel szimmetrikus a szűrő,Csak fele annyi együttható kell(a0...a127-->a0...a63). FRCT=1 legyen! Előtte bufferek feltöltése. Legelőször minden minta=0.

Menete: (Minden timer IT-re) **1**:bufferek update-je (új minta beolvasás, írás belejük cirkuláris címzéssel. Ez 2db AR -írópointerrel. Olvasó pointerek kezdeti értéke=író pointerek.) Xa-ba rendesen, Xb-be fordított sorrendben pakoljunk! **2**. Nullázzuk B akksit. **3**. N/2-szörös ciklusba tegyük: rptz B,#N firs xa,xb,együttható. **4**.Eredmény aBH-regiszterben.

●Adaptív szűrő: LMS x,y. utasítással.

●Polinom érték-számítás: x-> P(x). Erre: POLY *ARn+ . Előtte A,B,T-t fel kell tölteni. A poly-t N-szer ismételjük. ARn:együtth. x->T, a0->A, a1->B. AH->eredmény.

●Csak ARn módosítása: MAR ARn%+...

●STRCD,SACCD,SRCCD-kódtábla,DADST,DSADT-viterbi...

10.) Makrók: A makrók definiálásának nem kell .text szekcióban lenniük.

●definiálás:

●használat (beillesztés): makrónév param1,param2.

```
makrónév  .macro param1, param2...
          -utasítások-
          .endm
```

A hívásnál és a megadásnál ne legyenek ugyanazok a paraméterek nevei. A megfelelő változók behelyettesítődnek a kódrészletbe.

11.) Fájlok beillesztése:

.INCLUDE fájlnév

● *.EQU fájlok: Makrók, és állandók megadása ezekben a fájlokban.. include-dal kell beilleszteni. Érdemes hardverfüggő, pcb-függő, projectfüggő equ fájlokat készíteni.

● *.ASM fájlok: assembly programrészletek, szubrutinok. Ezt is incl.

Mintaprogram (template):

```
        .mmregs
        .sect ".ugrotabla"
        b inic
        nop
        nop
        b nmi_isr
        ...
        .sect ".programkezdet"
inic:   ssbx intm
        stm #itmaszk,imr
```

a gyári nevek használtához
ezt a hw reset vektorra kell linkelni
reset vektor. ugrás a kezdésre.
    távtartó a 4-gyel odébb
    lévő vektorhoz
első it vektor
többi it vekt. hasonlóan
ezt bárhova linkelhetjük
glob it tiltás
egyenként it eng/tilt

```
        rsbx cpl          | dp relatív címzés
        rsbx ovm          | nem szaturál az akksi
        ssbx sxm          | előjeles számokkal dolgozunk
        ssbx frct         | előjel kezelése szorzásnál
        stm #szam,pmst    | adatlapból a szamot +határ.
                          |   (új it vekt,ram,rom,clkout...)
        stm #szam,clkmd   | osc. idozites+inditas(pllndiv=1)
        stm #200,sp       | stack pointer
        stm #74924,swwsr  | 7 i/o wait state
          ...             | prifériák inicializációja
        rsbx intm         | glob it eng vissza
        b foprog          |
nmi_isr: ...              | első it szubrutin
          ...             |
         rete             |
           ...            | többi it rutin
foprog:  ...              | program utasítások, végtelen
           ...            |                    ciklusban
         b foprog         |
        .end              |
```

Regiszterek, inicializálás:

●.mmregs kell, hogy használhassuk az mmr-ek neveit.

●**MMR-ek**: 0000h-tól 001Fh-ig: IMR-interrupt maszk reg.(1-eng), IFR-megszakításforrás jelzése. ST0-0.státuszregiszter(TC-bit teszt eredménye, C-carry, OVA,OVB-overflow flag, DP-datapage p.), ST1-1.státuszreg(BRAF-block repeat aktív, CPL-relatív címzéshez, XF-láb állapota, HM-hold mode-holdban leáll e a proci, vagy csak lekapcsolódik a külső buszokról, INTM-global it eng-0engedve van, OVM-owerflow mode-1-határol-0nem, SXM-előjelkiterjesztés-1-on, C16-0-dupla precíziós mód-1-32bites mód, FRCT-szorzás utáni előjelkezelés-1-legyen! de resetre 0, ASM-shift érték)., PMST-processzor státuszregiszter (IPTR-it vektorok báziscíme, MP/MC-uP vagy uComp mód állítás, OVLY-ram overlay, AVIS-kijusson-e a címbusz az ic-n kívülre-1-kijut,DROM-rom látható-e az adatterületen, CLKOFF-kimenő órajel 1-re letiltva, SMUL, SST-szaturáció beállítás), AL, AH, BL, BH(08h...0Ch:**2db akksi** alsó/felső részek), AG, BG (akksik felső kiterjesztése:guard bitek. 8db), T, TRN-átmeneti regiszterek, AR0...AR7-indirekt címző regiszterek, SP-stack pointer, BK-cirkuláris buffer regisztere, BRC, RSA, REA-ciklusszámlálók, XPC-progmem cím kiterjesztő reg.

●**Az előjelkiterjesztés**: Az akku felső üres bitjei: 0-k legyenek, vagy az előjelet másolja minden helyre a proci.

●**IT**: vektorok :bázis+offset. Bázist az PMST.IPTR -rel lehet beállítani. Default: FF80. Offszetek: Reset=0, ... Ha elállítjuk, akkor a neki szánt ugrótábla-szekció elé tegyünk: .align -hogy lapháron kezdődjön. Ügyanis csak oda szabad tenni!!!!! Az üres helyeket .space n*16 beírásával töltsük ki!

●**hova tegyük**: MP-mód: fix ff80-ra linkeljük. MC-mód: bárhova linkelhetjük a ramban, a bootloader odateszi. ff80-ra nem lehet, mert ott rom-van! Ahova tesszük, oda kell állítani inic.-kor az iptr-t!

●Megszakítás elfogadás után global IT tiltva lesz, IACK-láb 0-ra vált. Visszatérés: RETE. Az egyes vektorok között 4sző az eltérés. Ugrótábla: mindegyikre egy B cím, utána 2db NOP. Üres helyekre: RETE.

Globális IT eng.:RSBX INTM. Tiltás: SSBX...

●**Külső IT**: INT0...iNT3, és NMI-aktív alacsony lábakkal. Visszajelzés: IACK -jellel. Ez is aktív alacsony.

Perifériák:

Timer:

Lefelé számlál, 16bites számláló (TIM). Előosztó: PSC-számláló-regiszter(rész) (csak 4bit!). Ezt a CLKOUT jel hajtja meg. Ha leszámolt, TINT-interruptot okoz. Honnan számoljon: PRD-regiszter. ezzel tint-kor updat-elődik a tim-regiszter. Ekkor íródik felül a psc is, a TDDR-regiszter(rész)ből. Ez az osztási arány. A psc, és a tddr a TCR-óravezérlő regiszter részei. Benne találóók még a TSS (1->óra stop), és a TRB (ha 1be állítjuk: tim, és psc update), FREE (1-szabadon fut), SOFT (1:stop 0-nál.0:azonnal) bitek. TOUT: T leteltekor jelző láb.

EBIF :

● (external bus IF, vagy EMIF:ext.mem.IF): A DSP adat, és címsine kimegy a lábakra: Progmem/IO/Adatmem közös adatbusz(16bit), címbusz(23bit). Aktuálisat a PS, DS, IS lábakon jelzi a dsp (negált jelek). A prog/data választást az MSTRB (negált) lábbal jelzi, az IO-t az IOSTRB(negált).Az adat és az i/ohoz tartozik R/Wjel is.

●Hogy működjön, a PMST.AVIS -bit=1 kell legyen.

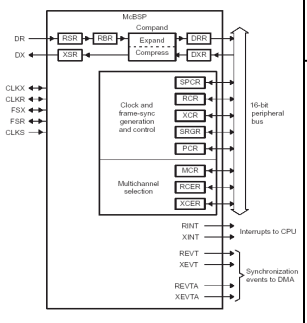
●Bankváltás: Ha kisméretű memória ic-ket használunk, azok közötti váltás időigényes. Ennek a kezelésére bankokra osztjuk a memóriaterületeket. Vezérlése: BSCR-regiszter (bankméret, kell-e késleltetés, külső IF-on/off...).

●Memória illesztése: 2fázisú a memóriakezelése (először a cím jelenik meg, majd a strobe-ra az adat is).

●Wait state generator: Lassú memóriák kezeléséhez beiktat wait-stateeket. Az SWWSR-regisztrerel állítható be: wait ciklusok külön az adat/prog alsó/felső térfélhez, és az iohoz. Max 7ciklus. Ha ettől is több várakozás kell, azt hardverből kell elintézni. Az utolsó sw várakozási ciklusban az MSC láb 0-ra vált. A hw várakozás vége: READY 1be váltásakor. Ha nem kell HW várakozás: READY=Vcc -re kötjük.

Soros port:

(SP0, SP1-standard sp., BSP, **McBSP**, TDM. A c5402-nek csak mcbsp-je van-több-csatornás bufferelt soros port) Ezzel a proci nélkül lehet kívülről írni/olvasni a belső memóriákat. 3 db van belőle: mcbsp0... mcbsp2. A jeleik végén is ott van a modul sorszáma. Szabványok:IIS, SPI, IOM-2, AC97... Max 128 csatornát kezel. 8-32bit.



●Használható standard, vagy bufferelt kommunikációra is. A kimenetek használaton kívül nagyimpedanciásak.

●A romban lévő táblázatok alapján képes ki/betömöríteni adatfolyamot.

●DX-adat adás láb, DR-vétel, adási/vételi keretszinkron jelek: FSX, FSR, bitszinkron jelek: CLKX, CLKR. Tartozhat megszakítás az adáshoz/vételhez: RINT, XINT. A DRR-be jön az adat, a DXR-pedig az adási adatregiszter. Mindegyikből 2 db van (DRR1, DRR2...a 32bites mód beállítási lehetőség miatt).

●**Beállítások**: (beállító és státuszbitek.) SPCR1(loopback, bitsorrend, előjel, órajelszinkonizálás, kim.eng, ITmód, error detect, vételi reg.full, vevő kész, vevő reset), SPCR2 (szabadonfutás, soft bit, keretszinkron reset, mintavételi gen.reset, adás IT-mód, adás errordetect, kim.reg.kiürült, adó kész, adó reset), PCR(XIOEN:kimenet sorosp.-vagy gpio-mód-0:sorosp., RIOEN:bemenetekre ugyanez, FSXM: frame sinc kívülről-vagy belülről jöjjön, CLKXM: adóórajelre ugyanez, CLKRM: vevőórára ugyanez, CLKS_STAT, DX_STAT, DR_STAT: gpio-módban a bemeneti lábak értéke, FSXP, FSRP: keretszinkr.aktív magas/akt.alacsony, CLKXP, CLKRP: bitórajelekre ugyanez)-regiszterekkel.

●A belső regisztereket báziscímhez adják meg, a bázis minden modulnál más: mcbsp0:39h, 1.mod:49h, 2.mod:35h. De a neveik nem mások: címtáblázatból használjuk. (subbankok)

●2 egség kommunikációjakor bármelyik adhatja a bitórajelet, de meg kell egyezni. Ha belső: a sample rate generator adja. A bitórajel felfutóélénél változik az adat, a lefütóra detektál.

●**komm. órajel**:A procióraje vagy a CLKS bemenet leosztva. Az SRGR1, SRGR2 -regisztrekkel (clk leosztás, fr.sinc. impulzus-szélesség, clk szinkronizálás ha külső clk-t használunk, él select, fő clk forrás választás, fr.sinc mód, hány leosztott clk-periódus 1 keret) lehet beállítani.

●**A keret** lehet 1-2 fázisú, a fázisokban lehet több szó(1-128), 8-32bites. A kommunikációt szavanként kell kezelni, és figyelni kell a keretszinkron jelet. De pl 4db 8bites szó=1db 32 bites, így csak egyszer kell a procinak kezelni fázisonként. Fázisonként külön megadható a szavak száma (pl ac97 audio codec szabványban sem egyforma a 2 f.).

●**Többcsatornás** mód: 1fázisúnak kell konfigurálni, és a fázisban annyi szó legyen, ahány egység akar csatlakozni a közös soros buszra. 1 szóhoz tartozó idő (időszelet, csatorna) 1 egységhez tartozik: (TDM). Mivel a kimenetek 3state-esek, nem csak pont-pont kapcsolat lehetséges. Max 32 egység, fix 128cstornahely. Egy egység használhat több csatornát is. Beállítások: MCR1(multichannel-on-off, használt/aktuális vevőcsatornák kiválasztása), MCR2 (ua. mint az előzőnél), XCERA/B, RCERA/B (csatornakiválasztás)-regiszterekkel.

●**SPI-mód**: clock stop módban megy: SPCR1.CLKSTP, PCR.CLKXP-vel lehet beállítani. A BCLKX, BDX, BDR, BFSX-lábakat használjuk spi-nek. Az spi-ben más nevek vannak, de nem baj.

●**GPIO-mód**: A CLKX, FSX, DX, CLKR, FSR, és DR-lábak használhatóak GPIO-ként. Ehhez a modulnak reset-ben kell lennie, és a PCR-ben be kell állítani az IOEN biteket. Az FSR lábon az FSRP bit értéke jelenik meg, ha kimenetként állítjuk be, ha bemenetként akkor innen olvasható a láb. A clk lábak hasonlóan. A PCR.DX_STAT a DX lábra, A PCR.DR_STAT- a DR-láb értéke.

● Auto buffer mód(ABU):A DMA-vezérlővel koprodukcióban műveli.

●**adc/codec** illesztésekor meg kell nézni a codec soros szabványát, az alapján állítsuk be az McBSP paramétereit.

●**Reset** után várni kell kicsit, a modul-startup-ra.

●RCR, XCR regiszterek: adás/vételi módok beállítása. Mindegyikből 2 db/modul van. (RCR1, RCR2...) (adás/vétel-kerethossz, szóhossz, tömörítés, fs-figyelés kezdete, adatkésleltetés, 1 vagy 2-fázisú keret-pl.mono/stereo,)

GPIO:

Összesen 2 láb: BIO-bemenet+IT forrás (feltételként használható ugráshoz), és az XF-kimenet (Az ST1.XF -bittel állítható.). + még a 3x6db McBSP-láb, és a 8db HPI-láb.

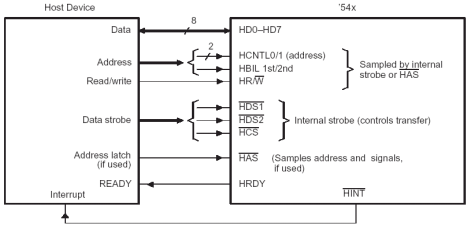
HPI:

8/16bites kibővített (EHPI) host-port interface. Host processor illesztési felülete a DSP-hez. A külső proci hozzáférhet a dsp memóriájához.

●**1.HPI-8**: Jelei: HD[0:7] (adatbusz), HCNTL[0:1] (címbusz), HBIL(alsó/felső bájt), HR/W, HDS1, HDS2 (strobe jelek.), HCS (chip select), HAS (ale-jel. ha nem haszn. 1-re kötjük), HRDY. A host vezérli a dsp-t, az pedig a readi-vel, vagy az adatbuszon válaszol. A HINT-jellel okozhat IT-t a hostnak.

●A komm.: A host címzéssel hozzáférhet a hpi-vezérlőregiszereihez: HPIC-vezérlő reg, HPIA-címregiszter, HPID-adatregiszter. Először beállítjuk a vezérlőt, beírjuk a címet, majd olvashatjuk az adatokat a dsp-ből. A HBIL-lábon adjuk meg, hogy a 16 bites adat alsó, vagy felső fele kerüljön a buszra. Az adatregiszter auto increment módon is hozzáférhető. A host megszakíthatja a DSP-t, a vezérlőreg megfelelő bitjével, de a dsp is a hostot, a hint lábbal. A 2 strobe bemenetet kössük össze:1strobe lesz.

●Hold: a host lekapcsolhatja dsp-t, a HOLD -láb lehúzásával. Ekkor a

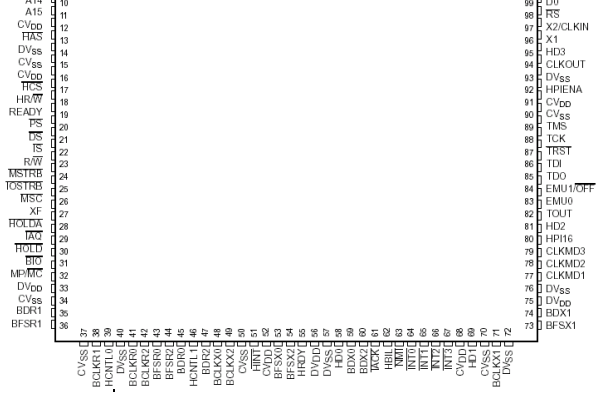


HOLDA-n visszajelez. Vissza: HOLD-ot elengedi a host.

●**GPIO**-kén használat: A HPIENA-lábat 0-ra kötjül. (Egyébként 1.) Ezzel a hpi működés tiltva van. A GPIOCR (lábak irányai), és a GPIOSR (lábak állapotai -R/W) regiszterekkel lehet bonyolítani a kommunikációt.

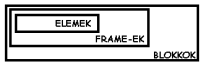
●**2: HPI-16**: 16 bites. +lábak: HA címbusz, HPIRS- modul reset, HPI16 -8/16bites mód választó láb (0: HPI8, 1:HPI16).

Bekötés:



DMA vezérlő:

Az EMIF, DARAM, EHPI, és a perifériák közötti pont-pont kapcsolatokra. Kivéve EHPI<->perifériák.



● IT,soros port, timer, cpu indíthat dma-transfert. 6db csatorna van.

A dmavez. is adhat it-t, ha átvitte a blokkot.

●módok: 1blokk, végtelen (auto-init).

●körforgásos prioritásosak a csatornák.

●külön memóriatérképe van, az alapján kell kinézni a címzést.

●külön dma-busz van, ezért nem zavarja a procit.

●beállítható cím management: auto increment, indexelés, lin/cirkobuffer...

●vezérlő regiszterek: DMPREC (fő vezérlő: emulációs beállítás, csatornák prioritásai, csatornák eng./tilt., kész átvitel milyen it-t okozzon a procinak), DMSA (subbank kijelölő címregiszter), DMSDN (subbank adatregiszter), DMSDI (u.a., csak ha ezt használjuk, akkor minden eléréskor inkrementálódik a dmsa).

●Minden csatornához 5db regiszter. Ezeket a dmsa, dmsi, dmsdi regisztereken keresztül lehet elérni, nem MMR-ek: DMSRCn: (Forráscím), DMDSTn(célcím), DMMCRn (mód: auto init, it generálás, multiketet/ABU-mód, it mikor, címzési mód, forrás/cél dmdm/pmem terület választás), DMCTRn(elemszámláló), DMSFCn: (framek száma/keret, sync select-mire induljon a transzfer).

Beírjuk a címreg-be a subaddress-t, aztán az adatregisztereken keresztül elérhetők a belső regiszterek.

●Címzési módok:

Multiketet: elemek, fram-ek, blokkok. Adattömeg másolásra.

ABU: auto buffer mód. Az McBSP adat ki/beviteli bufferelésére. Ekkor az egyik cím auto-módosítás nélküli, a másik pedig pl auto increment.

Rendszer:

●RS: reset láb. aktív alacsony.

Bootolás:

A dsp indításkor külső forrásból tölti be a programot, **ha MC** módban van. (MP módban nincs boot, FF80 (a progkezdet) Források: HPI, EEPROM (8/16bites párhuzamos), i/O, soros boot, melegindítás.

●**Menete**:A belső ROM-nak adódik át a vezérlés, ahol a bootloader program van. Beolvassa az FFFF-I/O címről bootolás módját (BRS-bájt), és kezdőcímét, (bootmem kezdőcím mód A mód:01-párh. 8bites boot) És aszerint tölti be a programot. A programcímeket a linker cmd fájlban adjuk meg a bootcímet a hardvertervező adja meg. A kezdőcím az, ahova az eepromunkat akarjuk tenni. Betöltés: a daramba tölti a pmem. területre. A loader az ovly=1 beállít.

●**Bootfájl**: Az eepromban spec boot-fájlt kell tölteni. Ebben bennevannak a szekciók, és címeik,és a programunk kezdőcíme is! Tehát akárhonnan indíthatjuk a programunkat, a linkerben oda tesszük, ahol nem szégyelljük. A bootloader oda helyezi. A kezdőprogram nevesített szekcióban legyen, vagy 1 text legyen.

●I/O boot: bio és xf jelekkel handshaking. +kell az adatbusz.

●soros boot: a soros porton keresztül.

●A melegindítás: programkezdet közvetlen megadása. külső pmem.

●HPI-boot. reset, vagy IDLE2 állapotban a host áttölti a programot.

Clock generátor:

A CLKMD1, CLKMD2, CLKMD3 -lábakkal lehet beállítani az órajelképzés módját: X2/CLKIN-bemeneti láb leosztva 2-vel, 4-gyel, vagy PLL-módok (fbe-t szorozza a pll .pl.001:10-zel szoroz. max 15x).

●Ha PLL módban van a dsp, akkor lehet szoftverből is beállítani a végleges órajelfrekvenciát. Persze nem muszály. A resetértéke a 3 módlábtól is függ. A CLKMD regiszterrel állíthatók be (osztásarány, szorzás, pll indítás). Indítás: (pll behúzás) A regiszter, PLLNDIV=1 -be állításával. Ezután megy csak a gyorsítás, kis időzítés (PLLCOUNTx16 clk ciklus)letelte után. A szükséges időt diagramról vegyük. Beállítjuk az időt, módot, bekapcs, vár, prog futtat. Ezután ne állítgassuk a pll-t!

●**Óra bekötés**: vagy külső oszcillátor az X2/CLKIN-lábra, vagy egy kvarckristály az X1, X2/CLKIN lábak közé. hozzá kersze 2 db kis kondi.

● f_{be}=10...20MHz lehet. Ennél a típusnál lehet kisebb is.

●Ha működés közben akarjuk változtatni a frekit, akkor előtte a PLLNDIV-vel kapcsoljuk le, átállít, pll vissza, vár, program futtat tovább.

●Az 5402-nél lábprogramozással beállítható teljesen a pll is, más DSP-knél nem: azoknál lábbal csak: pll/pll nélkül, lepsztva -lehet.

Tápellátás:

●2 feszültségről megy: i/o: DVdd=3,3V core: CVdd=1,6V. Startup: Előbb az i/o fesznek kell felmennie, ha már jó, akkor indulhat a core fesz bekapcsolása. Külön van a föld bemenet is: DVss, CVss.

●Spórolás: IDle-módok (power down). IDLE 1, IDLE 2, IDLE 3 -utasításokkal. Ezekből resetre, vagy külső megszakításra tud ébredni a proci. IDLE 1-ben (Idd>10mA) csak a proci áll le, IDLE 2-ben (Idd>2mA) a perifériák is, IDLE 3-ban a clk is. IDLE 1-ből belső IT-re is tud ébredni.

●Még: HOLD mód. A host lehúzza HOLD-lábat, a dsp visszajelez a HOLDA-lábon, és hold állapotba kerül. Ekkor Idd->75%.

J-tag:

● Ehhez a TDI, TDO, TMS, TCK, TRST, TCK_RET, EMU0, EMU1. A vezetékek a 14 lábú j-tag-csatlakozóhoz max 6inch-esek legyenek. Az emu0, emu1, tms, tdi lábakat felhúzóellenállásra kell kötni.