Aplicații în CCS

Sim28xx: timer, PIE

## -Proiect Procesoare de Semnal-

# Ce este un Procesor Digital de Semnal (eng. Digital Signal Processor -DSP)?

Un DSP este un microcontroler special conceput pentru aplicații de procesare de semnal. Cele mai uzuale operații utilizate în procesarea de semnal sunt convoluția, filtrarea și conversiile în domeniul timp-frecvență. Acestea au nevoie de adunări și înmulțiri executate în mod recursiv. Alfel spus, se realizează operații de multiplicare și acumulare (eng. multiply and accumulate -MAC). Microprocesoarele standard execută operațiile de înmulțire ca și adunări recursive ceea ce implică un număr excesiv de adunări în cazul operațiilor MAC. Astfel, procesele pot avea timpi destul de mari de execuție. Spre deosebire de acestea, DSP-urile conțin unități MAC speciale care pot executa aceeași operație într-un singur ciclu mașină. De exemplu, un DSP cu 150 MIPS poate procesa aproximativ 32 de milioane de eșantioane de date pe secundă în comparație cu un microprocesor standard cu 150 MIPS care poate procesa numai 2 milioane de eșantioane pe secundă. [1]

# Aspecte generale

Simulatorul C28x simulează arhitectura completă a setului de instrucțiuni C28x (cu excepția instrucțiunilor de emulare).

În ceea ce provește memoria, simulatorul oferă o simulare configurabilă a acesteia. În mod implicit, el nu face nicio mapare în memorie specifică dispozitivului sau procesorului. Toate blocurile din spațiul de Program, Date și Intrări/Ieșiri pot fi simulate prin adăugarea unori blocuri de memorie folosind fișiere de configurare specifice. Blocurile de memorie pot fi, adăugate, de asemenea, folosind funcția „Memory Map Add” din meniul GEL.

Simulatorul realizează o simulare precisă a ciclului mașină atât pentru blocuri de bază, cât și pentru blocuri de memorie. De exemplu, pentru memorii on-chip, simulatorul reglează numărul de cicli necesari pentru a accesa memoria în funcție de stările de așteptare specificate în fișierul de configurare.

Simulatoarele F2810 și F2812 modelează următoarele periferice:

* PIE –Peripheral Interrupt Expansion Block (întreruperile ce țin de periferice)
* Flash
* Timer
* VBUS

PIE, Timer-ul și memoria Flash sunt disponibile ca module suplimentare, în timp ce VBUS face parte din nucleul simulatorului C28x. Pentru a simula un periferic, acesta trebuie inclus în fișierul de configurare al simulatorului.

Instrumentul Pin Connect permite utilizatorului să simuleze evenimente din lumea externă. În această privință, cele mai interesante pentru utilizator sunt semnalele de control provenite de la entitățile externe ale simulatorului. Simulatorul oferă utilizatorului o listă cu pinii corespunzători diferitelor semnale de control. Utilizatorul trebuie să specifice momentele de timp la care au loc evenimentele ce urmează a fi declanșate pe pinul respectiv, listându-le într-un format special într-un fișier. Următorul pas este crearea unei legături între acest fișier și pin folosind comanda window, comenzi GEl sau Pin Connect.

Simulatorul C28x oferă funcția Pin Connect pentru toate configurațiile procesorului. Lista de pini acceptate este prezentat în Tabelul 1.

**Tabelul 1. Pini acceptați de funcția Pin Connect din simulatorul C28x**

|  |
| --- |
| *INT1* |
| *INT2* |
| *INT3* |
| *INT4* |
| *INT5* |
| *INT6* |
| *INT7* |
| *INT8* |
| *INT9* |
| *INT10* |
| *INT11* |
| *INT12* |
| *INT13* |
| *INT14* |
| *DLOGINT* |
| *RTOSINT* |
| *NMI* |
| *EMUINT* |

# Funcția Port Connect permite utilizatorului să simuleze transferul de date între DSP și o entitate externă care este prezent în hardware-ul real. Transferul se poate face în două moduri:

* Date de la entitatea externă către DSP;
* Date de la DSP înspre entitate externă.

Pentru a simula transferul de date de la o entitate externă la DSP, mai întâi, toate valorile de date care vor fi transferate de la entitatea externă la DSP sunt introduse într-un fișier. Apoi se face o asociere în simulator între acest fișier și adresa la care se află entitatea externă. Această asociere se numește read-mode port-connect. Ori de câte ori simulatorul trebuie să citească de la entitatea externă prin adresa asociată, acesta citește câte un cuvânt din fișier. Pentru a simula transferul de date de la DSP la o entitate externă, un fișier este conectat la port în modul de scriere împotriva adresei entității externe. Ori de câte ori simulatorul trebuie să scrie date la adresa respectivă, acesta scrie câte un cuvânt în fișier.

Simulatorul C28x oferă facilități de conectare a porturilor pentru zona de program, zona de date și intrri/ieșiri pentru toate configurațiile procesorului. Are instrucțiuni specializate, cum ar fi IN, OUT și UOUT pentru a executa operații de intrare/ieșire, scriere și citire. Este important aici ca adresa la care se realizează conexiunea să fie deja mapată în memorie (simulatorul nu include în mod implicit aceste adrese în harta sa a memoriei). În cazul porturilor seriale, datele pot fi transmise prin conectarea unor fișiere la locațiile mapate de memorie pentru registrul de transmisie a portului serial în modul de scriere. În mod similar, datele pot fi primite prin conectarea unor fișiere la locațiile mapate de memorie pentru registrul de recepție a portului serial în modul de citire. [2]

C28x este un dispozitiv protejat în ceea ce privește pipeline-ul. Deși hardware-ul garantează execuția corectă prin introducerea standului de cicluri, rezultatul poate fi obținerea mai multor cicluri de execuție decât este necesar, ceea ce face ca programul să fie mai puțin eficient în ceea ce privește performanţa. De asemenea, nivelul maxim de optimizare posibil este limitat în prezent de capacitățile compilatorului. După optimizarea maximă este realizată de aplicația DSP a compilatorului, optimizarea suplimentară este o provocare. Asemenea hardware-ului, simulatorul C28x introduce cicluri de blocare pentru a evita pericolele de date. Deoarece smulatorul este o reprezentare software a dispozitivului, este posibil să se utilizeze flexibilitatea acestuia pentru a dezvolta un cod mai eficient. [3]

# Timer-ul

C28x conține 3 timere identice pe 32 de biți (timer0, timer1 și timer2) care au perioade presetate și suportă divizarea semnalului de ceas. Aceste 3 timere au un un registru cu numărătoare pe front descendent care generează o întrerupere când numărătorul ajunge la zero. Numărătorul este resetat automat cu valoarea perioadei.

Timere-ele pot seta/comuta pe un pin de ieșire atunci când numărătorul ajunge la zero. Timer2 este rezervat pentru sisteme de operare proiectate pentru aplicații în timp real (eng. Real-Time Operating System – RTOS) și este conectat la întreruperea de bază INT14 a simulatorului. Timer0 și Timer1 pot genera, de asemenea întreruperi atunci când sunt conectate la întreruperile PIE (Peripheral Interrupt Expansion).

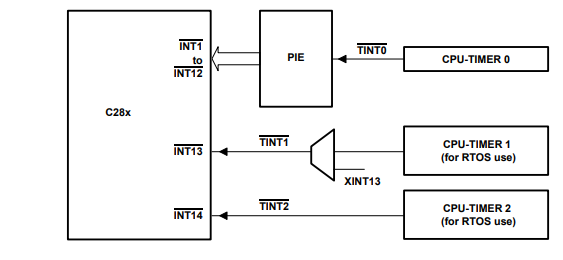


Figura . Întreruperile de la Timer (fig. extrasă din TMP320F2810PGFS)

# Pentru a configura simulatorul pentru a utiliza perifericul Timer, sunt adăugate următoarele module în fișierul de configurare al simulatorului:

*//////////////////////////////////////////////////////////  
// Modul Timer //  
//////////////////////////////////////////////////////////*

*module main;  
cssi\_modules timer0,timer1,timer2;  
end main;*

*module timer0;  
cssi\_library Timer.dll;  
init\_function initTimer;  
index 0;  
pinc ..\\tutorial\\sim28xx\\timer\\timer0.dat;  
end timer0;*

*module timer1;  
cssi\_library Timer.dll;  
init\_function initTimer;  
index 1;  
pinc ..\\tutorial\\sim28xx\\timer\\timer1.dat;  
end timer1;*

*module timer2;  
cssi\_library Timer.dll;  
init\_function initTimer;  
index 2;  
pinc ..\\tutorial\\sim28xx\\timer\\timer2.dat;  
end timer2;*

Parametrul „index” (0, 1 sau 2) este necesar pentru utilizarea timer-ului și specifică care timer este utilizat.

Parametrul „pinc” este opțional și specifică pinul conectat la fișierul de ieșire și, respectiv, la ce timer vor fi activate întreruperile. Un fișier tipic pinc are următoarea structură:

*85  
136  
187  
238*

Acest lucru arată că timer-ul specificat a numărat în jos până la zero la ciclurile 85, 136, 187 și 238. Se poate determina astfel că acest timer a avut o perioadă de 51 de cicluri.

Fișierul pinc va fi suprascris de fiecare dată când este solicitat simulatorul.

Când se utilizează perifericul Timer (Timer.dll) cu simulatorul, nu trebuie specificate în fișierul de configurare registrele Timer-ului mapate în memorie(0xc00-0xc3f). Timer.dll gestionează aceste registre intern. Dacă fișierul de configurare mapează această memorie, atunci Timer.dll nu va afișa valorile numărătorului în registrele mapate în memorie.

În continuare se urmărește rularea pas cu pas a unui program în limbaj de asamblare care utilizează toate 3 timer-ele. Timer0 și Timer1 nu sunt conectate la perifericul PIE, așa că atunci când numără pe pantă descendentă, nu vor genera nicio întrerupere. Numai Timer2 va genera o întrerupere de bază INT14 în acest exemplu. Programul în limbaj de asamblare va configura timer0 să aibă o perioadă de 50 de cicli, timer1 cu o perioadă de 100 de cicli iar timer2 cu o perioadă de 200 de cicli. Timer2 va ilustra utilizarea prescalerului.

Folosind Windows Explorer, navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\timer. Deschidem fișierul timer.gel în Notepad și căutăm o funcție numită ***StartUp()*** în acel fișier. Înlocuim numele funcției cu ***TutorialFunction()*** și salvăm fișierul.

O imagine care conține text

Descriere generată automat

Dublu-click pe icon-ul Setup CCStudio v3.1 apoi apăsăm pe butonul ***Remove all*** pentru a elimina configurațiile anterioare. Acest proces trebuie realizat, chiar dacă anterior s-a folosit tot Simulatorul F28xx. Aceasta va reseta proiectul. Facem click dreapta pe pictograma F28xx Simulator Tutorial din lista configurațiilor disponibile din panoul central, click pe Add to System apoi Save și Exit din meniul File. Confirmăm deschiderea CCStudio în fereastra care apare ulterior.

O imagine care conține text

Descriere generată automat

Dacă există mai multe instalări ale CCStudio, se ***selectează Open→F28xx Simulator Tutorial / CPU în CCS: Parallel Debug Manager.***

Facem clic pe Proiect→Deschidem și navigăm la C:\CCStudio\_v3.10\tutorial\sim28xx\timer.

Selectăm timer.pjt și facem click pe Open.

Facem click pe File→Load Program și navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx \timer\Debug. Selectăm timer.out și dăm click pe Open.

Click pe Click View→Registers→CPU Registers.

O imagine care conține masă

Descriere generată automat

Click pe File→Load GEL și navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\timer. Selectăm timer.gel și dăm click pe Open. Acest fișier GEL va adăuga registrele timer-ului mapate în memorie la fereastra ceasului.

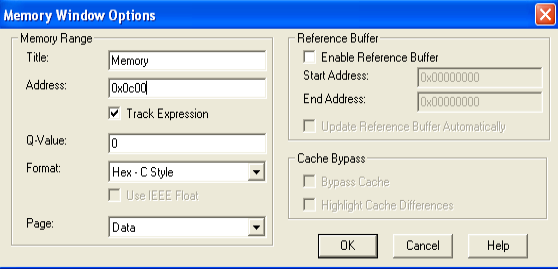
O imagine care conține text

Descriere generată automat

Din meniul View alegem opțiunea GEL Toolbar. În caseta de comandă apărută, tastăm TutorialFunction() și apăsăm butonul Execute Command().



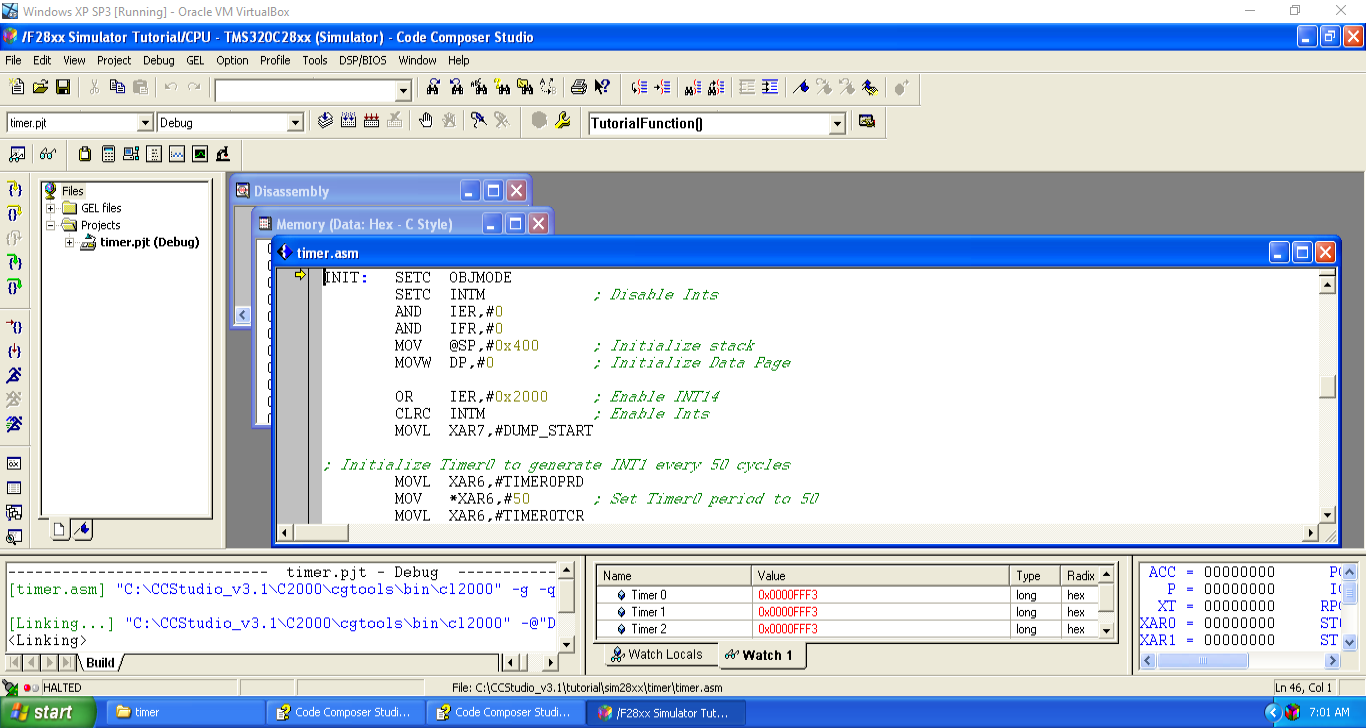
Facem click pe View→Memory. Introducem 0x0c00 în câmpul de adresă (Address) și ne asigurăm că câmpul pagină (Page) are selectat ‘Data’.



O imagine care conține text, masă

Descriere generată automat

Click pe Debug®Reset CPU.



În continuare se rulează programul pas cu pas (Single-step) până când PC este la eticheta LOOP.

O imagine care conține text

Descriere generată automatÎn acest moment ar trebui să observăm timer0 (\*0xc00), timer1 (\*0xc08), și timer2 (\*0xc10) cum se decrementează.

O imagine care conține text

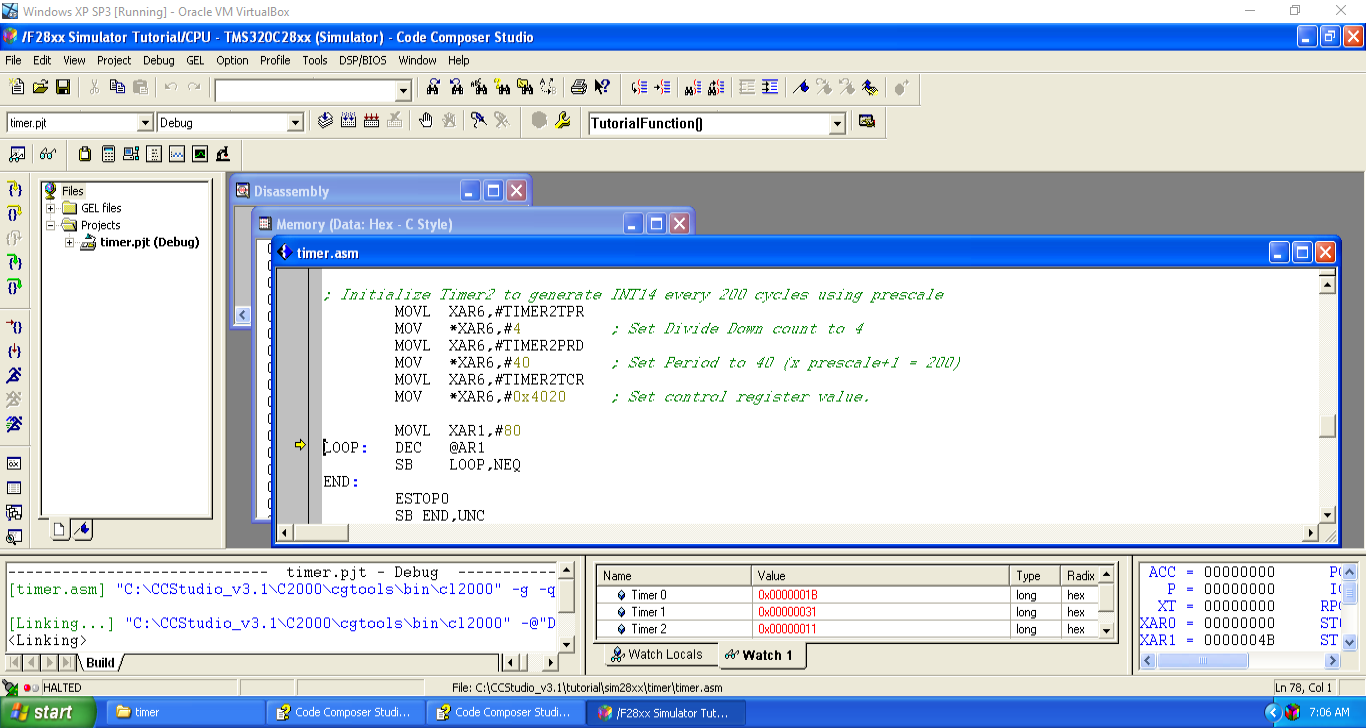
Descriere generată automat

Single-step până când observăm timer0 (\*0xc00) că trece prin zero.

O imagine care conține text

Descriere generată automat

Ar trebui să efectuăm doar 2 pași. Se poate urmări cum programul reîncarcă timer0 cu 0x32 și continuă numărătoarea inversă. Cu alte cuvinte, valoarea sa se mută de la un număr mic, cum ar fi 0x9, la un număr mai mare, cum ar fi 0x2B.



Single-step până când observăm timer2 (\*0xc10) că trece prin zero.

Aceasta durează câțiva pași. În acest moment ar trebui să vedem cum IFR = 0x2000, indicând faptul că întreruperea INT14 este în așteptare.

Continuăm rularea pas cu pas prin rutina de tratare a întreruperii INT14 (eng. Interrupt Service Routine – ISR).

Rulăm programul până la finalizare. Programul se va opri la instrucțiunea ESTOP0.

Facem click pe File®Open. În lista care apare alegem All Files (\*.\*). În continuare selectăm timer 0.dat, timer1.dat, și timer2.dat și facem click pe Open.

O imagine care conține text, tablă albă

Descriere generată automat

O imagine care conține masă

Descriere generată automat

O imagine care conține masă

Descriere generată automat

Ar trebui să vedem că timer0 s-a executat la fiecare 50 de cicluri, timer1 s-a executat la fiecare 100 de cicluri și timer2 s-a executat la fiecare 200 de cicluri. Dacă numărul de cicluri este puțin mai mic, este posibil ca o instrucțiune cu mai multe cicluri să fi fost executată atunci când numărătorul a ajuns la zero, sau programul a fost executat dintr-o memorie aflată într-o stare de așteptare diferită de zero. (A se vedea fișierul de configurare a simulării C:\CCStudio\_v3.10\drivers\sim28xxtutorial.cfg.)

***799-748 = 51***

***311-210 = 101***

***551-346 = 205***

Din meniul File al CCStudio, alegem Exit pentru a închide programul. Este important să ne asigurăm că restaurăm fișierul de configurare original al simulatorului când am finalizat cu acest tutorial.

# 4. PIE

PIE multiplexează 96 de surse de întreruperi în două întreruperi de bază (INT1 și INT2). Fiecare întrerupere de bază suportă 8 surse individuale. Când PIE este activat, vectorii din aceste 96 de întreruperi, plus întreruperile de bază care rămân sunt aduse din Tabela vectorilor de întreruperi care rezidă la adresa 0xd00-0xdff din spațiul de program.

Pentru a configura simulatorul să utilizeze perifericul PIE, este nevoie de adăugarea următoarelor module în fișierul de configurare al simulatorului.

*//////////////////////////////////////////////////////////  
// Modul PIE//  
//////////////////////////////////////////////////////////  
module main;  
cssi\_modules Pie;  
end main;*

*module Pie;  
cssi\_library Pie.dll;  
init\_function initPIE;  
int3 . .\\tutorial\\sim28xx\\pie\\int3.dat;  
int9 . .\\tutorial\\sim28xx\\pie\\int9.dat;  
int27 ..\\tutorial\\sim28xx\\pie\\int27.dat;  
int29 ..\\tutorial\\sim28xx\\pie\\int29.dat;  
int81 ..\\tutorial\\sim28xx\\pie\\int81.dat;  
end Pie;*

Pie.dll este perifericul PIE pentru simulator, situat în folderul C:\CCStudio\_v3.10\drivers. Programul folosește fișiere de conectare pentru pini specificând întreruperile PIE (int1 ... int96). Aceste fișiere au același format și funcție ca cele utilizate de plug-in-ul Pin Connect.

Timer0 și timer1 ar putea genera, de asemenea, întreruperile PIE. Pentru a utiliza timere pentru a semnaliza o întrerupere PIE, trebuie doar să înlocuim calea către fișierul de conectare a pinului cu „timer0” sau „timer1”.

Când se utilizează perifericul PIE (Pie.dll) cu simulatorul, nu trebuie specificate în fișierul de configurare registrele PIE mapate în memorie(0xce0-0xdff). Pie.dll gestionează aceste registre intern. Dacă fișierul de configurare mapează această memorie, atunci Pie.dll nu va deservi întreruperile PIE.

În continuare, se urmărește rularea unui program simplu în limbaj de asamblare care inițializează toate întreruperile de bază și vectorii PIE. Toate rutinele de tratare a întreruperilor înregistrează ID-ul lor de întrerupere într-o locație de memorie (0x8000), făcând mai ușor să vedem ce întreruperi au avut loc și în ce ordine. Fișierele de conectare pentru pini vor genera următoarele întreruperi:

***80 int9.dat*** *generează întreruperea de bază INT2****130 int3.dat*** *generează întreruperea de bază INT1****180 int27.dat*** *3 întreruperi apar simultan, 2 cu aceeași întrerupere de bază (core INT)****180 int29.dat*** *pentru a afișa gestionarea corectă a priorităților****180 int81.dat*** ***250 int29.dat***  *generează nucleul INT4 pe o sursă cu prioritate mai mică*

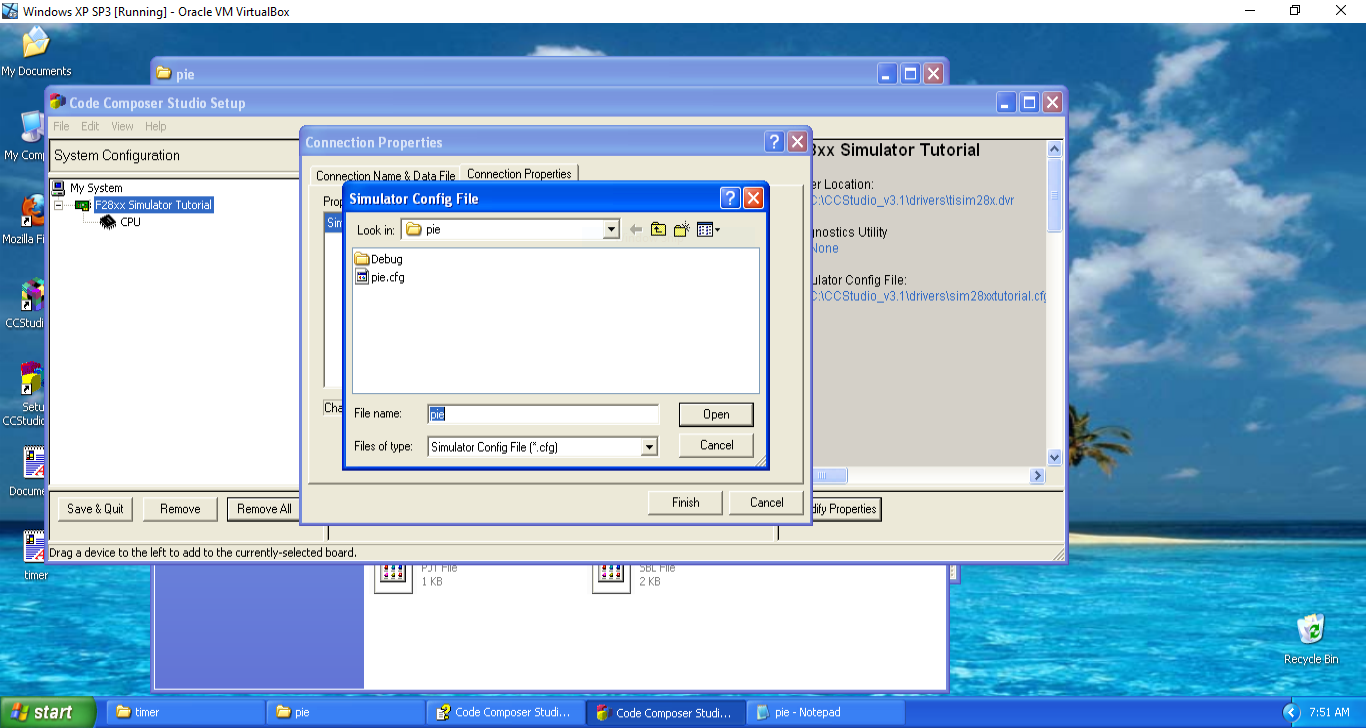
***260 int27.dat***înainte de apariția unui INT4 de bază pe o sursă cu prioritate mai mare.

Folosind Windows Explorer, navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\pie. Deschidem fișierul pie.gel în Notepad și căutăm o funcție numită StartUp() în acel fișier. Înlocuim numele funcției cu TutorialFunction() și salvăm fișierul.

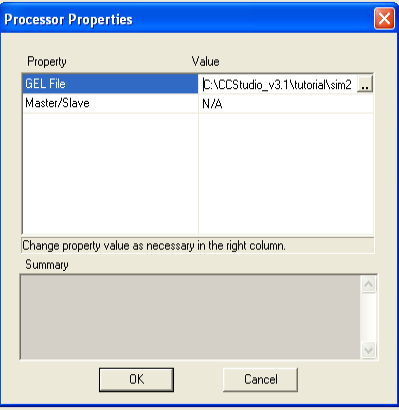
O imagine care conține text

Descriere generată automat

Dublu-click pe icon-ul Setup CCStudio v3.1 apoi apăsăm pe butonul Remove all pentru a elimina configurațiile anterioare. Acest proces trebuie realizat, chiar dacă anterior s-a folosit tot Simulatorul F28xx. Aceasta va reseta proiectul. Facem click dreapta pe pictograma F28xx Simulator Tutorial din lista configurațiilor disponibile din panoul central, apoi click pe Add to System. Facem click dreapta pe pictograma simulatorului F28xx din panoul System Configuration din partea stângă și selectăm Properties din meniul contextual. Se va afișa fereastra Connection Properties. Facem click pe butonul Next pentru a vizualiza fila Conection Properties. Facem click pe butonul Browse de lângă opțiunea Simulator Config. Navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\pie. Selectăm pie.cfg și facem click pe Open. Acționăm butonul Finish pentru a salva modificările.



Facem click dreapta pe pictograma F28xx Simulator Tutorial 🡪simulator CPU procesor din panoul de configurare din stânga și selectăm Properties din meniul contextual. Se va afișa fereastra Processor Properties. Accesăm butonul Browse de lângă opțiunea GEL File. Navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\pie. Selectăm pie.gel 🡪Open🡪OK🡪Save și Exit din meniul File.



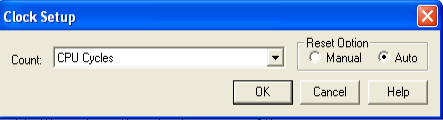
Selectăm Project🡪Open și navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\pie. Selectăm pie.pjt 🡪Open. Dăm click pe File 🡪Load Program și navigăm în folderul C:\CCStudio\_v3.10\tutorial\sim28xx\pie\Debug și deschidem fișierul pie.out.

Din meniul View alegem opțiunea GEL Toolbar. În caseta apărută tastăm TutorialFunction() și apăsăm butonul Execute Command().

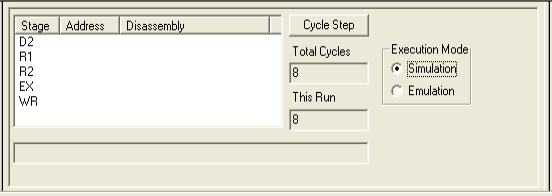


Click pe View 🡪 Registers 🡪 CPU Registers.

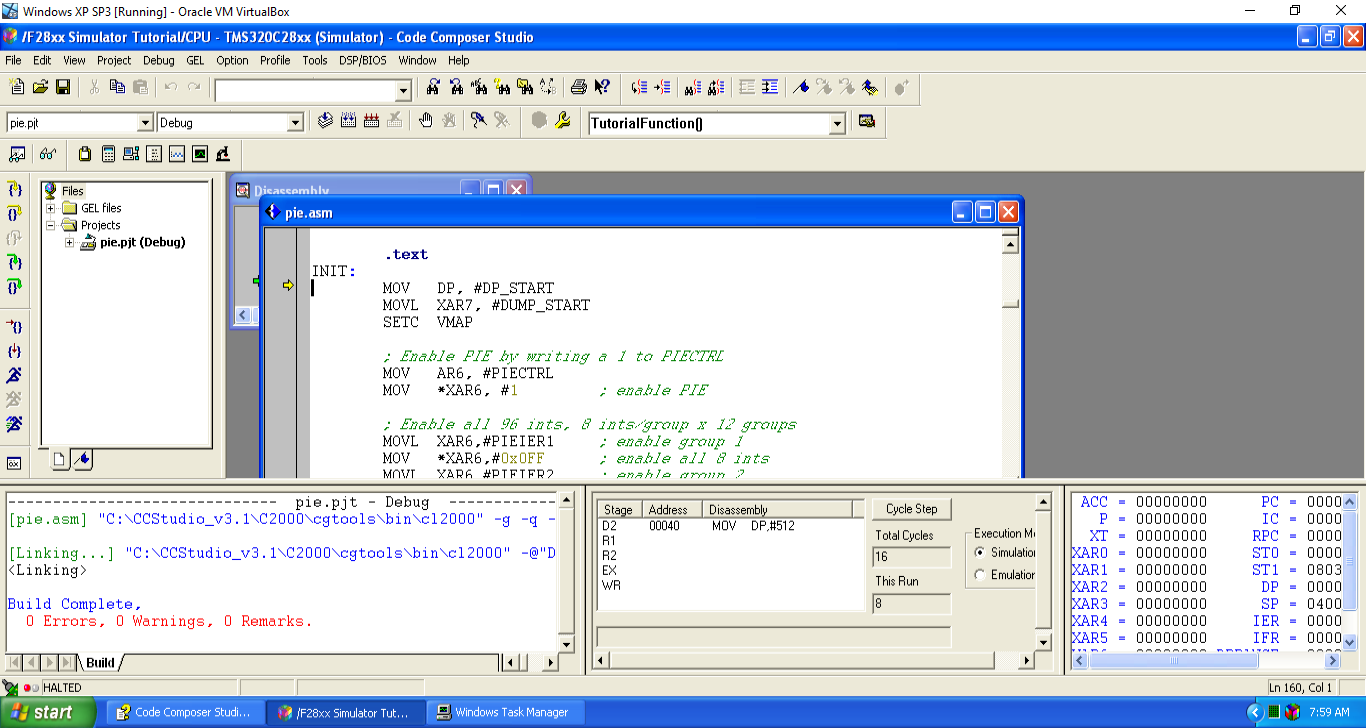
Click pe Profile 🡪Clock 🡪Setup. Selectăm Auto ca și Opțiune de resetare și confirmăm prin apăsarea butonului OK.



Facem click pe Tools 🡪C28x Pipeline Display. În fereastra de dialog, selectăm butonul radio Simulation din secțiunea Execution Mode.



Click pe Debug 🡪Reset CPU.



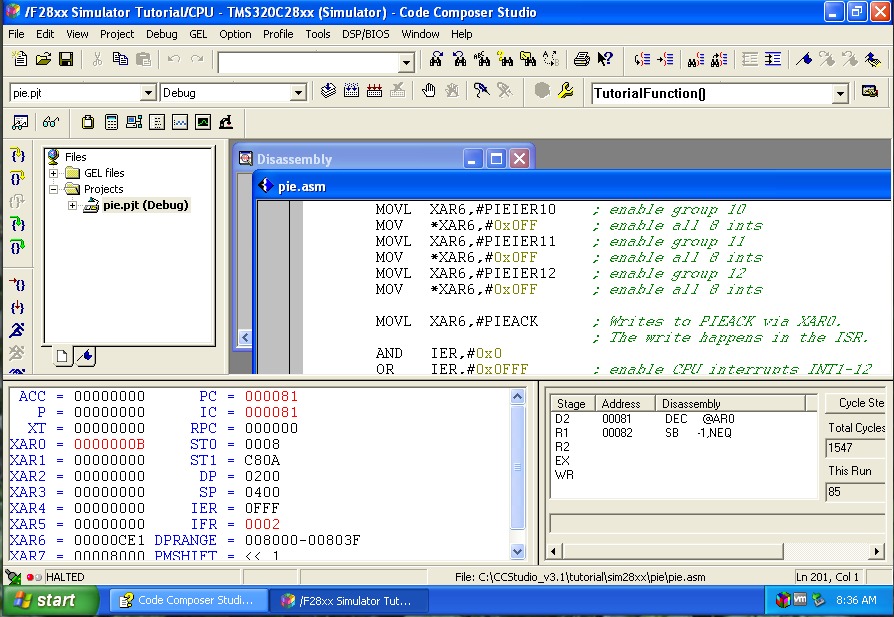
Rulăm pas cu pas de câteva ori. Trebuie să vedem câmpurile Total Cycles și This Run cum se incrementează în fereastra Pipeline Display. Câmpul This Run reprezintă numărul de cicli necesari pentru a executa instructiunea în limbaj de asamblare, așa că se poate să nu fie schimbat pentru fiecare instrucțiune.

O imagine care conține text

Descriere generată automat

Rulăm pas cu pas până când câmpul Total Cycles atinge o valoare de 80-90.

Prima întrerupere PIE (int9) va avea loc în jurul celui de-al 80-lea ciclu. Aceasta se poate observa lângă cursorul din fișierul pie.asm. Rulăm încet pas cu pas până când IFR devine egal cu 2. IFR = 2 chiar înainte de a intra în PIE IFR = 2 chiar înainte de a intra în PIE ISR int9.



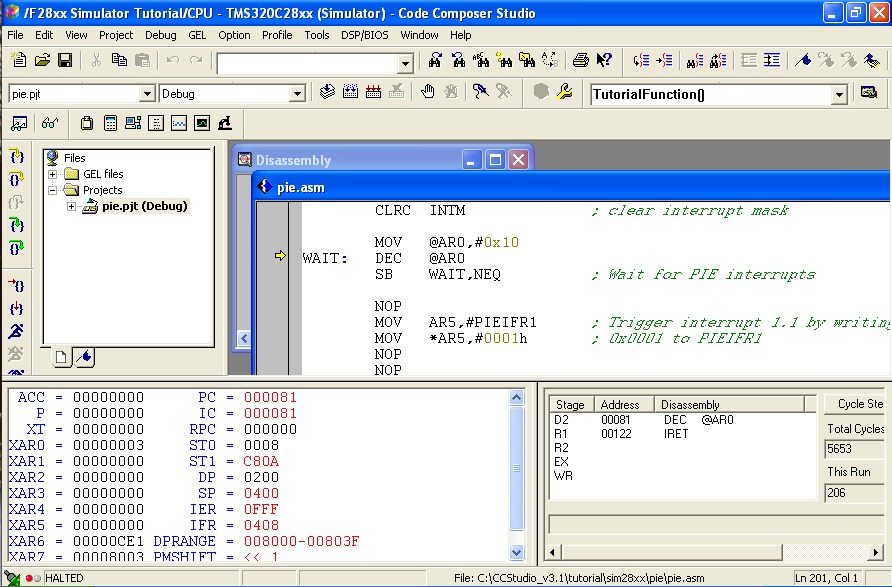
Rulăm pas cu pas prin ISR până când ne întoarcem înapoi în programul principal.

Rulăm pas cu pas până când câmpul Total Cycles atinge o valoare de 130-140.

A doua întrerupere PIE (int3) are loc pe la 140 de cicli. Ar trebui să putem observa IFR=1 chiar înainte de a accesa rutina de tratare a întreruperii PIE int3.

Rulăm pas cu pas prin ISR până când ne întoarcem înapoi în programul principal.

Rulăm pas cu pas până când câmpul Total Cycles atinge o valoare de aproximativ 190. Trei întreruperi PIE (int27, int29 și int81) vor apărea simultan la 190 de cicluri. Putem observa IFR = 0x408 chiar înainte de a intra în ISR PIE int27.



Putem verifica dacă toate cele 3 întreruperi sunt în așteptare uitându-ne la locațiile mapate de memorie pentru IFR-uri PIE la 0xCE9 și 0xCF7. Acestea ar trebui să conțină 0x0014 și, respectiv, 0x0001.

O imagine care conține text

Descriere generată automatO imagine care conține text

Descriere generată automat

Rulăm pas cu pas prin PIE ISR int27 până când ne întoarcem în programul principal.

Facem încă o singură etapă pentru a intra în ISR pentru următoarea întrerupere în așteptare (int29).

Rulăm pas cu pas prin PIE ISR int29 până când ne întoarcem în programul principal.

Mai facem un pas pentru a intra în rutina de tratare a întreruperii int81.

Rulăm pas cu pas până când ne întoarcem în programul principal. Acum toate întreruperile în așteptare au fost deservite.

Mai facem 2 pași până când câmpul Total Cycles atinge aproximativ 260. Următoarea întrerupere PIE (int29) va apărea la aproximativ 260 de cicli urmată de întreruperea int27 la 290 de cicli. Ar trebui să se poată observa IFR=8 chiar îniante de a accesa rutina de tratare a întreruperii int29. Rulăm pas cu pas prin aceasta până când ne întoarcem în programul principal.

Rulăm încă un pas pentru a intra în rutina de tratare a întreruperii int27 până când ne întoarcem în meniul principal. Următoarea întrerupere ce urmează a fi deservită va fi generată de programul principal și indicată prin registrul de flaguri IFR1 (Intterrupt Flag Register).

Întreruperea nu va apărea decât cu 5 instrucțiuni (5 cicli) înainte de a scrie în PIE IFR1.

Facem rulare pas cu pas prin ISR int1 până când ne întoarcem în meniul principal.

Facem rulare pas cu pas în programul principal până la instrucțiunea OR IFR,#2. Acest ultim exemplu va demonstra că preluările vectorilor de bază (0x000000-0x00003f) apar atunci când PIE este dezactivat și VMAP = 0. Mai facem un pas să accesăm isr int2. Facem rulare pas cu pas prin ISR int2 pentru a termina exemplul.

Click view 🡪Memory. Introducem 0x8000 în câmpul Adresă și ne asigurăm că câmpul Pagină are Data selectat, apoi facem clic pe OK.

O imagine care conține text

Descriere generată automat

# Bibliografie

|  |  |
| --- | --- |
| [1] | International Islamic University Islamabad, „Digital Signal Processing LAB-experiment No. 5: Introduction to CCS and DSP Kit”. |
| [2] | Texas Instruments, „TMS320F2810, TMS320F2812 System Control and Interrupts Reference Guide,” *SPRU712G,* 2009. |
| [3] | T. Instruments, „TMS320C28x INSTRUCTION SET SIMULATOR,” *SPRU608,* 2003. |

[4] www.ti.com/ccstudiodemos