Sisteme cu F. P. G. A. și D. S. P.

– Metode de programare a platformei DSP TMS320 - F28069M LaunchPAD –

I. INTRODUCERE:

Platforma C2000 TMS320 F28069M LaunchPAD a fost concepută, pentru a fi compatibilă cu mai multe medii de programare și dezvoltare de aplicații. În cadrul acestui document vom aborda trei metode principale prin care platforma F28069M LaunchPAD poate fi programată, iar modul de execuție al programului realizat poate fi testat în timp real, anume:

- mediul de dezvoltare și programare Code Composer Studio IDE;
- mediul de simulare și testare Matlab – Simulink + Code Composer Studio (versiunea 3.3);
- mediul de simulare și testare SolidThinking Embed / Altair Embed / VisSim;

Pentru a verifica funcționalitatea programului scris, pe lângă platforma f28069m LaunchPAD, vom utiliza câteva componente electronice în vederea verificării funcționalității unităților periferice consacracte în procesarea digitală de semnal (intrări și ieșiri digitale, generatoare de semnal modulat în lățime, comparatoare, convertoare analog – digitale).

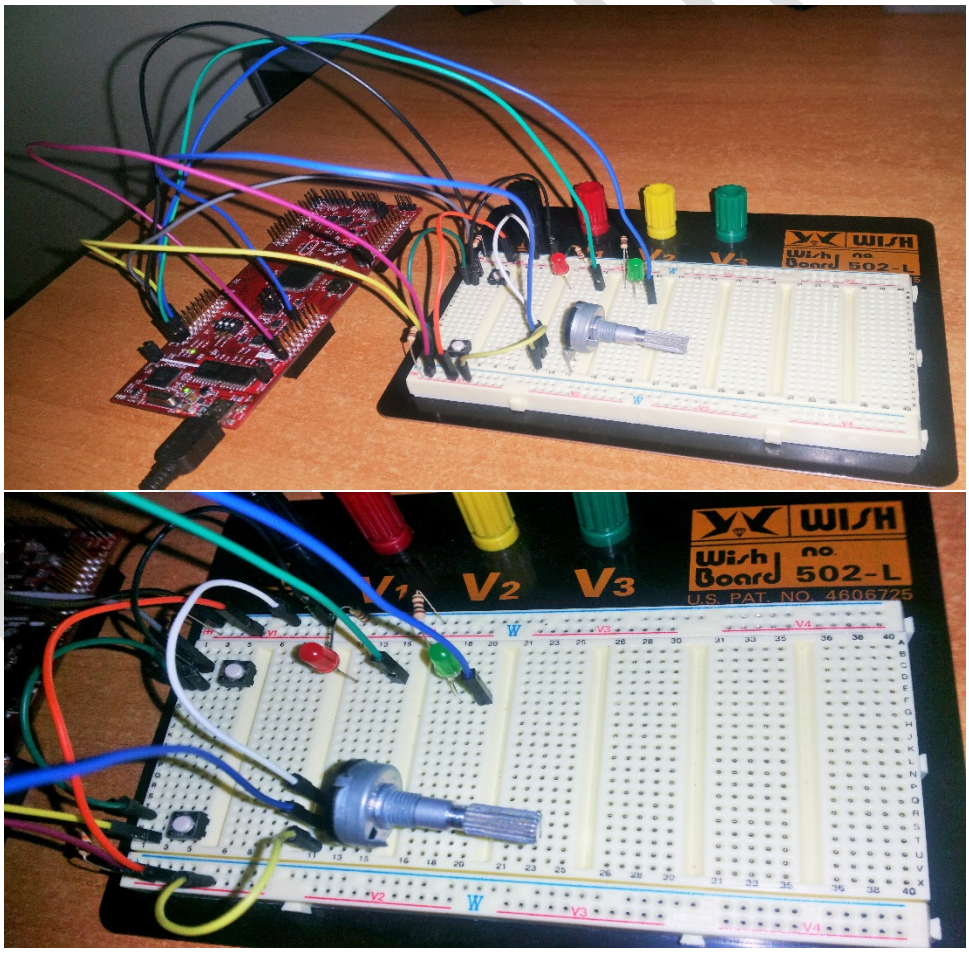


Fig. 1 – Montaj experimental

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Montajul experimental conține următoarele componente:

Componentă	Denumire pin	Rol pin	Dispozitiv testat
Comutator cu acționare prin apăsare și revenire	GPIO 19	Intrare digitală	Comparator
Comutator cu acționare prin apăsare și revenire	GPIO 44	Intrare digitală	Comparator
Diodă electroluminiscentă roșie	GPIO 0	Ieșire digitală	Generator de semnal
Diodă electroluminiscentă verde	GPIO 1	Ieșire digitală	Generator de semnal
Potențiometru	ADCIN A0	Intrare analogică	Convertor analog - digital

II. METODE DE PROGRAMARE:

A. Medode manuale pentru generare a proiectului și a codului program::

1. Code Composer Studio (versiunea 5) – reprezintă mediul de dezvoltare și programare al tuturor platformelor cu DSP și micro-controller concepute de Texas Instruments. În cadrul acestui mediu, pot fi implementate atât aplicații specifice procesorului platformei de dezvoltare, cât și aplicații specifice calculatorului gazdă, în vederea stabilirii comunicației dintre cele două dispozitive. Limbajul de programare utilizat, poate fi „C" standard sau „C++", limbaj de asamblare, sau așa-zisa combinație „mixed assembly-syntax".

Pentru a realiza o aplicație în acst mediu se vor parcurge următorii pași:

Pasul 1 – Definirea spațiului de lucru – se realizează prin intermediul ferestrei din figură:

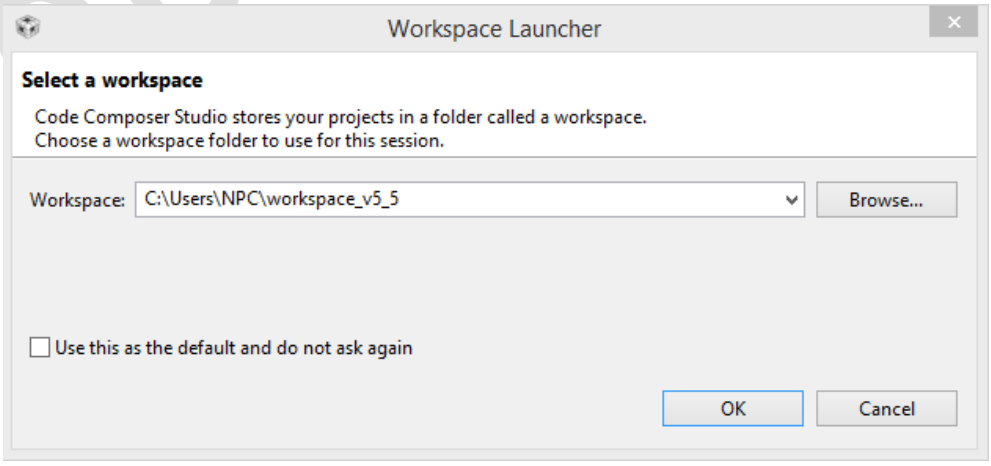


Fig. 2 – Definirea spațiului de lucru în mediul de dezvoltare Code Composer Studio

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

emd.utcluj.ro

Odată, lansat în execuție mediul Code Composer Studio, are două variante de dispunere a interfeței grafice: modul perspectivei de depanare și modul perspectivei de editare. Inițial, Code Composer Studio pornește în modul perspectivei de editare. În cadrul acestei perspective se definesc următoarele ferestre sau zone importante:
- fereastra de navigare în proiect (eng. Project Explorer) (marcaj verde);
- consola de stare (eng. Status Console) (marcaj albastru);
- zona de editor (eng. Editor Area) (marcaj portocaliu);
- zona pentru configurarea platformei de testare (eng. Target Configuration) (marcaj mov);

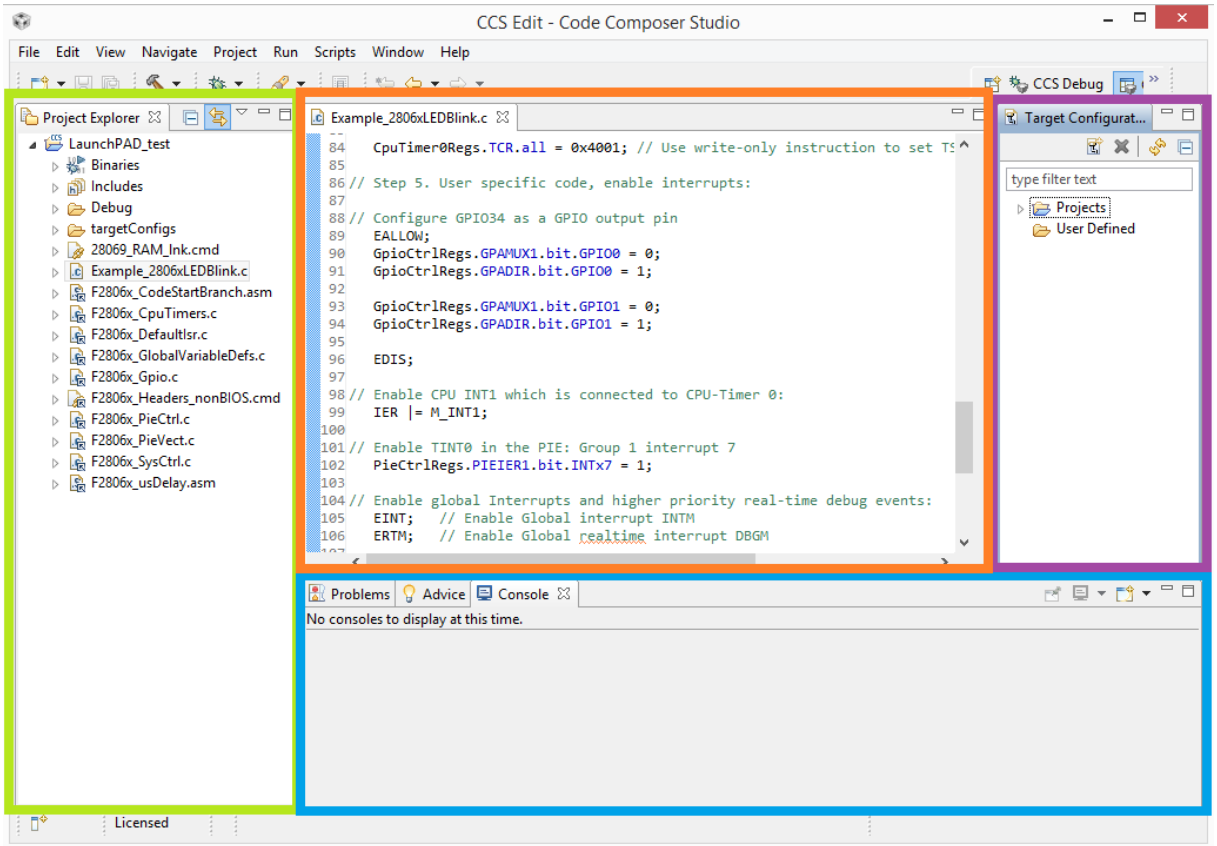


Fig. 3 – Mediul Code Composer Studio în modul perspectivei de editare

Conceperea unei aplicații în mediul Code Composer Studio, presupune crearea unui proiect nou. Proiectele, sunt compuse din mai multe tipuri de fișiere, prin care se constituie aplicația binară. Respectivele fișiere componente ale proiectului se împart în trei categorii:
- bibleoteci de funcții și fișiere sursă (ex. .lib sau .c);
- fișiere antet de definire (ex. .h);
- fișiere cu instrucțiuni de asamblare (ex. .asm):

Aplicația executabilă binară, se generează în urma procesului de compilare, și are extensia (sau formatul) „.out”. Respectivul fișier executabil, va fi încărcat fie în memoria RAM a platformei de dezvoltare (dacă se dorește doar testarea programului), fie în memoria program de tip FLASH (dacă se dorește inscripționarea permanentă a programului în memorie). Destinația aplicației executabile se decide pe parcursul dezvoltării. În continuare se propune, realizarea unui program pentru controlul ieșirilor digitale. Astfel:

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pasul 2 – Deschiderea unui nou proiect: În meniul „File” se alege „New” → „CCS Project”:

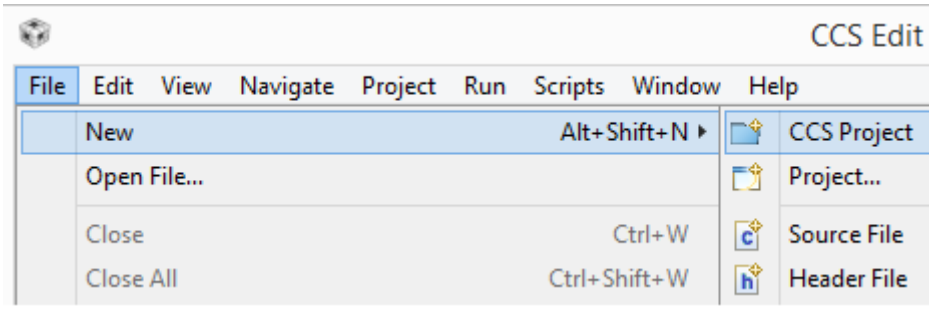


Fig. 4 – Deschiderea unui nou proiect în Code Composer Studio

Pasul 3 – Inițializarea proiectului: În fereastra nou deschisă, se vor completa următoarele aspecte privitoare la tipul proiectului ales:

- Numele proiectului (ex. test_f28069m);

- Tipul fișierului generat (eng. Output type) (ex. Executable);

- Utilizarea locației implicite pentru stocarea proiectului (eng. Use default location);

- Familia procesorului digital de semnal (eng. Family) (ex. C2000);

- Varianta constructivă (eng. Variant) (ex. prima căsuță – 2806x Piccolo – a doua căsuță – TMS320F28069);

- Tipul conexiunii jTAG cu calculatorul gazdă (eng. Connection – ex. Texas Instruments XDS100v2 USB Emulator);

- Șablonul standard al proiectului (eng. Project templates and examples) (ex. Empty Project – proiect nou gol);

Pentru a finaliza etapa de inițializare a proiectului nou, se va alege opțiunea „Finalizare” (eng. Finish), prin intermediul butonului de control al operației de inițializare;

epe.utcluj.ro

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

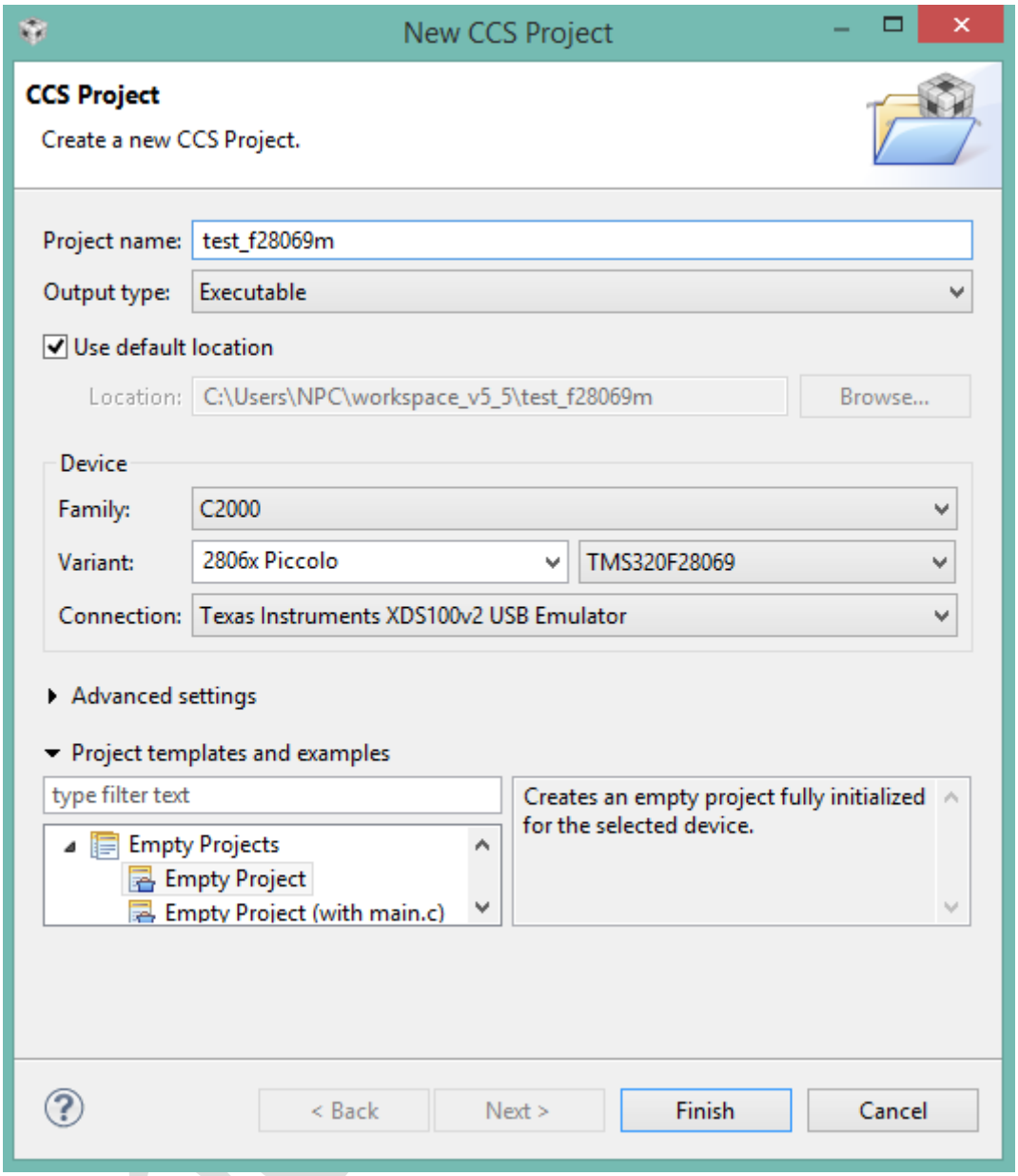


Fig. 5 – Fereastra pentru inițializare a proiectului

În urma procedurii de inițializare, în cadrul mediului de programare Code Composer Studio, mai precis a ferestrei de navigare în proiect, se vor creea în mod automat, structurile de bază ale unui proiect pentru Code Composer Studio, având aplicabilitate directă pe o platformă fizică (eng. hardware) de dezvoltare, precum C2000 TMS320F28069M LaunchPAD.

Structurile de bază ale proiectului presupun următoarele categorii de fișiere:

- fișiere antet (bibleoteci) specifice platformei F28069M (eng. Includes);
- fișiere specifice modului de configurare al platformei (eng. targetConfigs);
- fișiere pentru crearea legăturilor între componentele proiectului (28069_RAM_lnk.cmd);

Pe lângă fișierele incluse în mod implicit, se vor mai adăuga câteva componente suplimentare, care fac parte din colecția de programe auxiliare ControlSUITE, care, permite utilizatorului, să acceseze bibleoteci, exemple și fișiere pre-concepute pentru gestionarea perifericelor platformei de dezvoltare (ex. GPIO, ADC, SPI, I^2C, UART, jTAG etc...).

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pasul 4 – Introducerea fișierelor suplimentare, necesare în vederea realizării aplicației:

Înainte de a include fișierele suplimentare, trebuie stabilit un obiectiv al aplicației care se dorește a fi implementată. Se propune deci, conceperea unei aplicații de bază, prin care, să se controleze ieșirile digitale „GPIO0” și „GPIO1” în vederea realizării unei secvențe de semnalizare intermitentă cu două diode electroluminiscente (eng. LED) în mod simultan.

În acest sens, se va proceda astfel:

- Se vor verifica, dacă variabilele de sistem au fost inițializate corect în ceea ce privește directoarele ControlSUITE, astfel: din meniul „Window” se va alege opțiunea „Preferences”. În cadrul ferestrei deschise, se vor alege următoarele opțiuni: General → Workspace → Linker Resources. În partea dreaptă în categoria „Defined path variables” ar trebui să fie definită variabila cu numele „controlSUITE” având calea de acces „C:\ti\controlSUITE”. Dacă nu există această variabilă, se va defini prin intermediul butonului „New”.

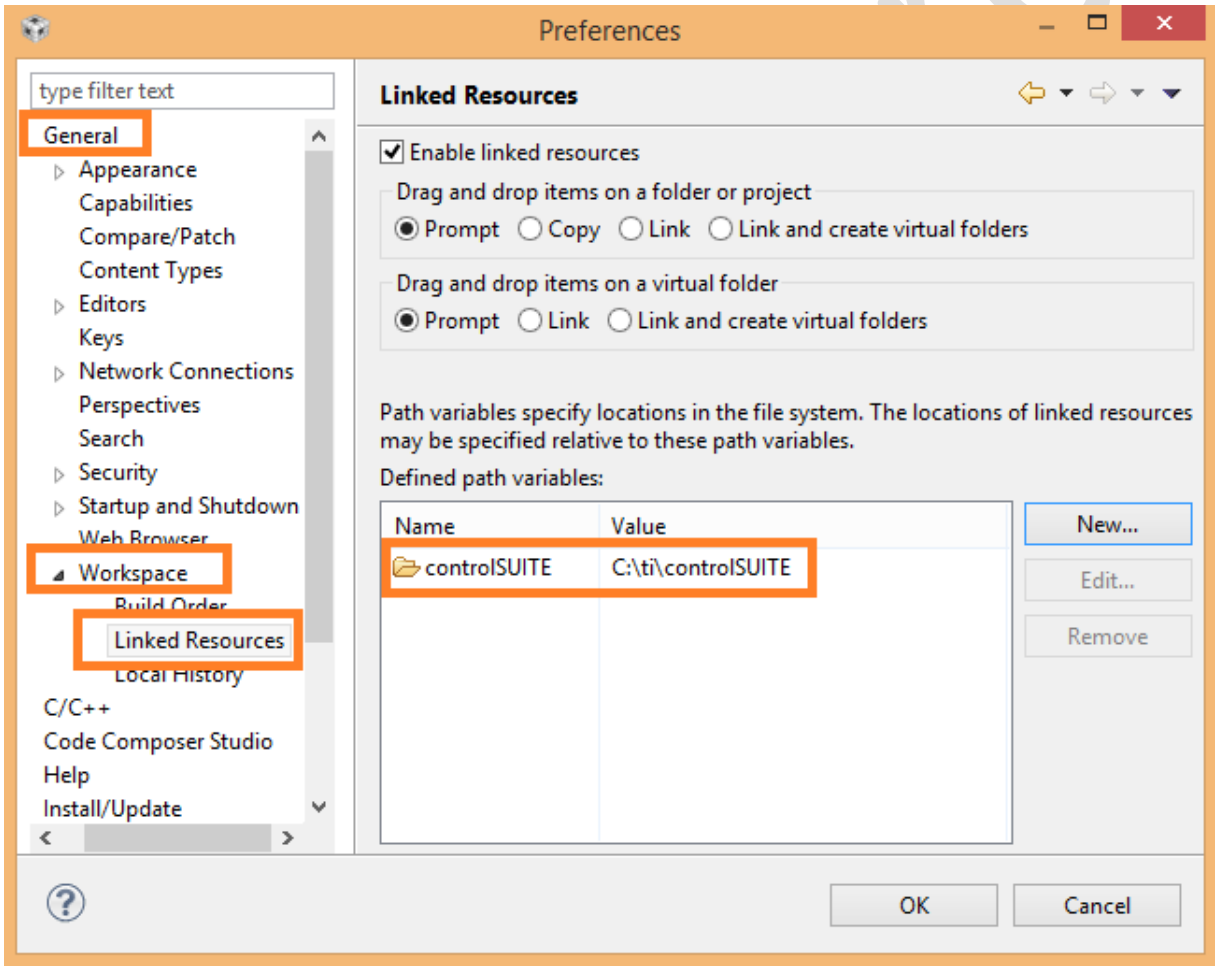


Fig. 6 – Verificarea variabilelor de sistem

În cadrul aceleiași ferestre în colțul din partea stângă sus, există o căsuță de căutare în care apare textul „type filter text”. În această căsuță, se va trece cuvântul „build”, iar apoi, din lista care se va genera în mod automat, se va alege opțiunea „Build Variables”. În mod similar, se verifică, faptul că, variabila „controlSUITE” este definită. În caz contrar, se va defini variabila ca și „Director” prin intermediul butonului „Add” (partea dreaptă, nu este afișat în figură!)

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

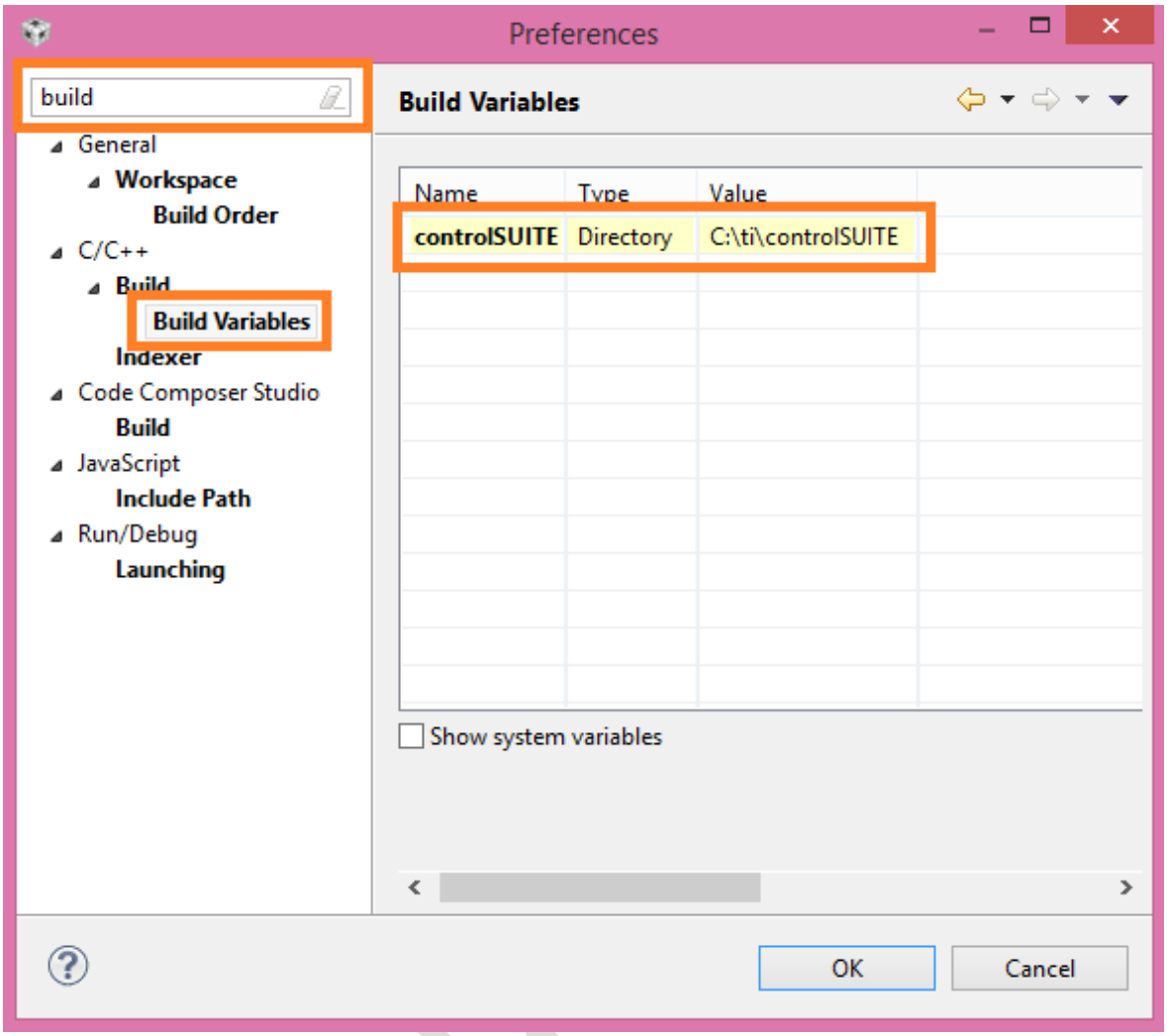


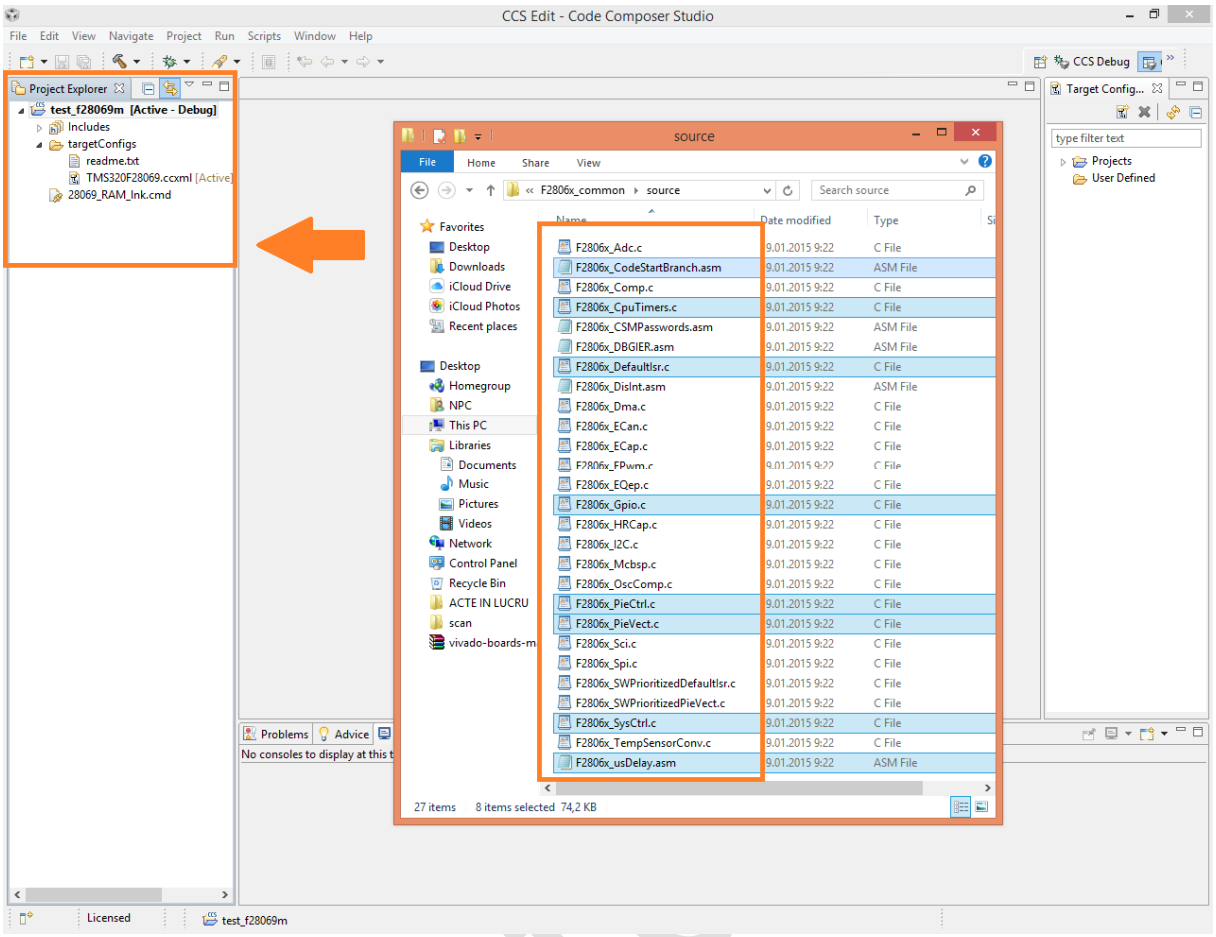
Fig. 7 - Verificarea variabilelor de sistem

Dacă operațiile anterioare au fost parcurse cu succes, se va finaliza întreaga procedură de verificare prin apăsarea butonului „Ok". Astfel, în cadrul proiectului, se va putea apela variabila cu numele controlSUITE, iar mediul Code Compose Studio, va identifica în mod automat faptul că, numele variabilei, face referire înspre calea „C:\ti\controlSUITE".

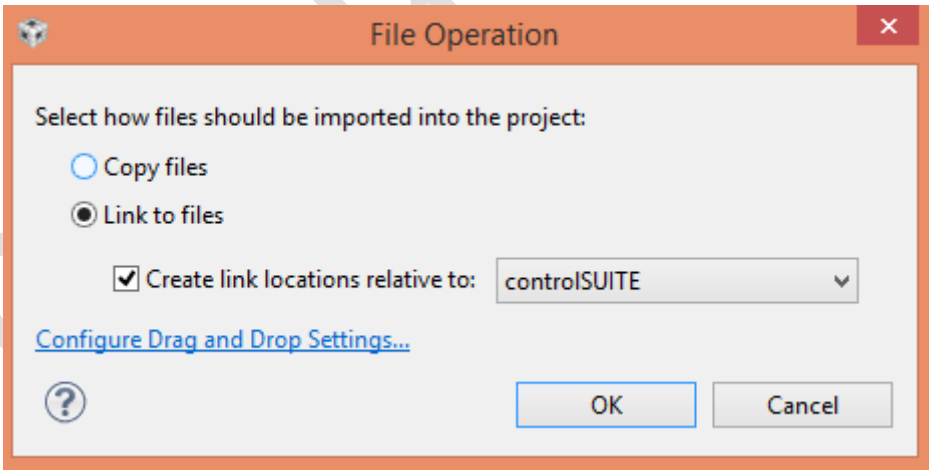
În continuare, se vor adăuga fișierele suplimentare, necesare aplicației pe care, dorim să o dezvoltăm în cadrul proiectului deschis. În acest sens se va proceda astfel:

- Cu ajutorul aplicației „My Computer / This PC" (Windows Explorer), se va naviga înspre calea: „C:\ti\controlSUITE\device_support\f2806x\v141\F2806x_common\source". Din cadrul acestor directoare, menținând tasta „Ctrl" apăsată, se vor alege următoarele fișierele: F2806x_CodeStartBranch.asm, F2806x_CpuTimers.c, F2806x_DefaultIsr.c, F2806x_Gpio.c, F2806x_PieCtrl.c, F2806x_PieVect.c, F2806x_SysCtrl.c, F2806x_usDelay.asm. Eliberând tasta „Ctrl" în urma selecției, fișierele vor fi preluate în Code Compose Studio, prin operația de deplasare efectivă, în spațiul de lucru, mai precis în navigatorul de proiect (Drag and Drop). Pentru a finaliza procedura, se va confirma în fereastra „File Operation" faptul că, se dorește crearea unei legături (eng. „Create link location relative to") cu variabila „ControlSUITE".

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro



A.



B.

Fig. 8 – A. Adăugarea fișierelor suplimentare în cadrul proiectului
B. Confirmarea adăugării fișierelor și crearea legăturii între fișiere și variabila ControlSUITE

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Se repetă aceeași procedură și în cazul fișierului „F28069x_GlobalVariableDefs.c” din: „C:\ti\controlSUITE\device_support\f2806x\v141\F2806x_headers\source”;
- Se repetă aceeași procedură și în cazul fișierului „F2806x_Headers_nonBIOS.cmd” din: „C:\ti\controlSUITE\device_support\f2806x\v141\F2806x_headers\cmd”;

Pentru a stabili o legătură între directoarele necesare proiectului și fișierele declarate, în vederea satisfacerii dependențelor impuse în proiect, se va proceda astfel:

- În cadrul ferestrei de navigare în proiect, se va efectua comanda „click dreapta” asupra titlului proiectului activ în desfășurare „test_f28069m [Active – Debug]”, iar din meniu, se va alege opțiunea „Properties” (Proprietăți);

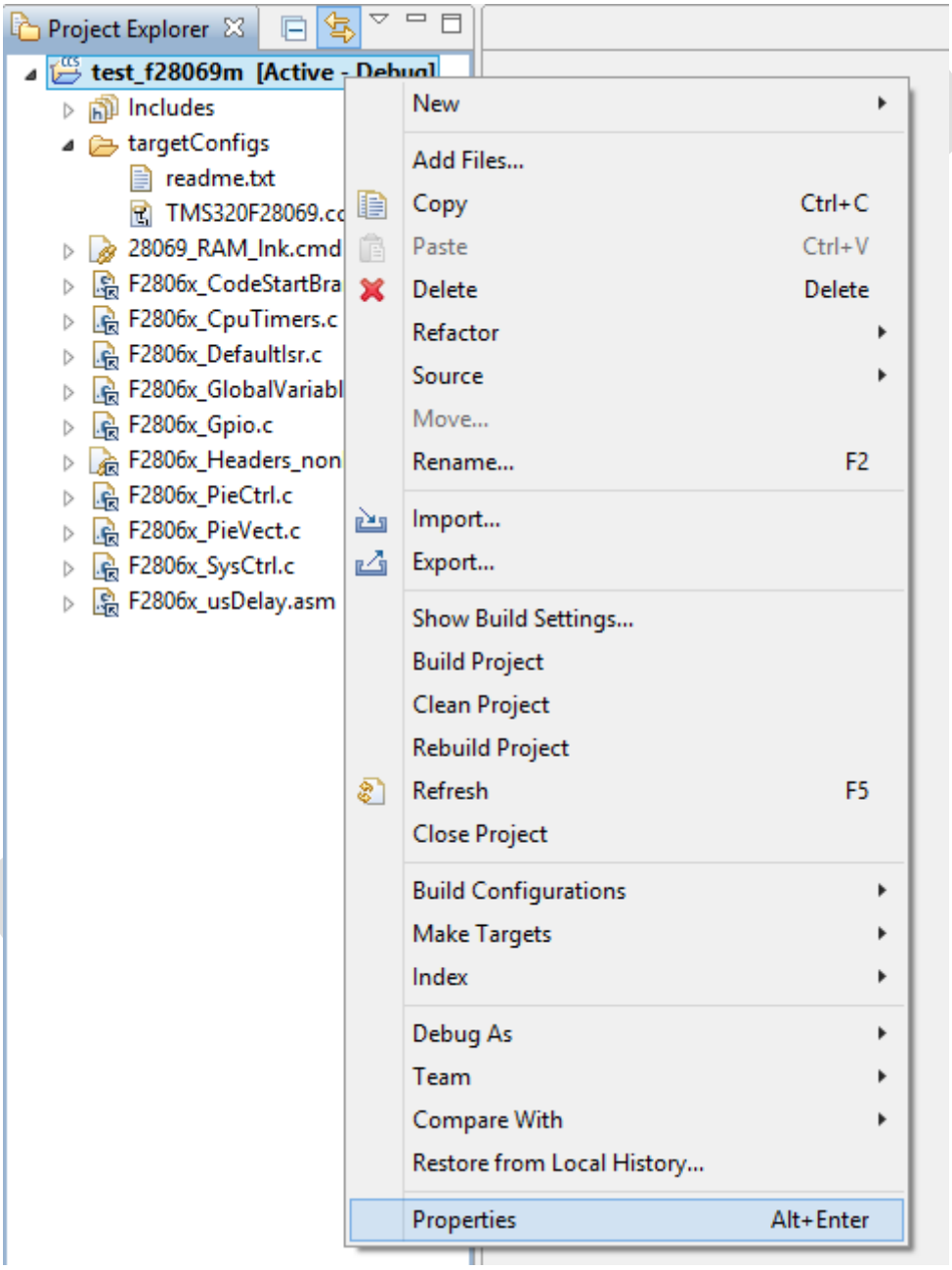
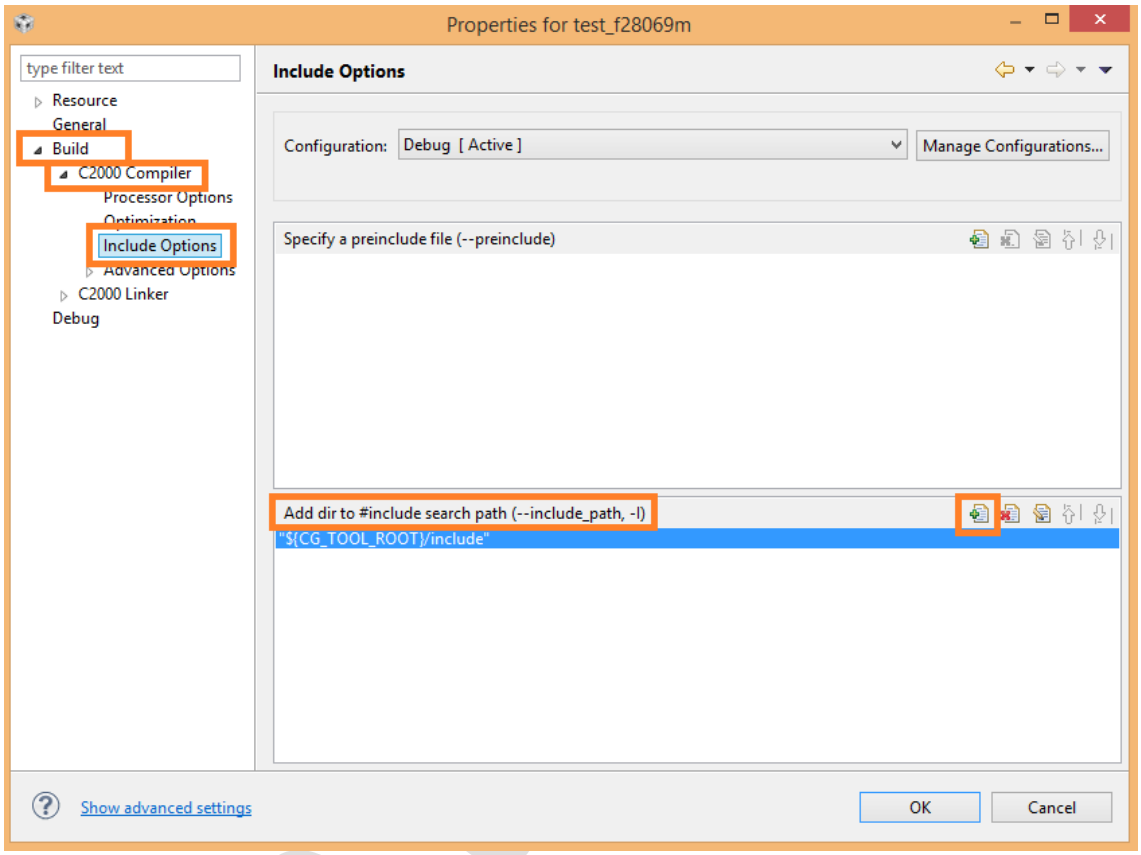


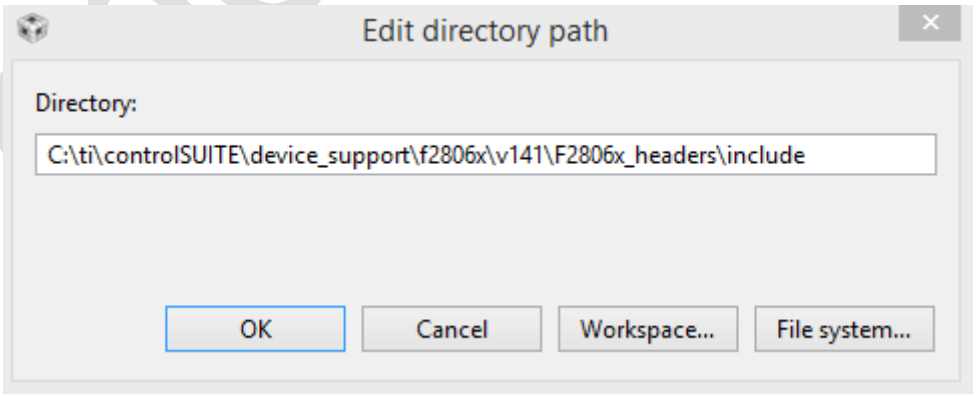
Fig. 9 – Deschiderea meniului „Properties” pentru particularizarea proiectului

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- În cadrul ferestrei „Properties for test_f28069m”, în categoriile „Build” → „C2000 Compiler” → „Include Options”, zona „Add dir to #include search path...”, cu ajutorul iconiței „fișier +” se vor adăuga următoarele directoare:
C:\ti\controlSUITE\device_support\f2806x\v141\F2806x_headers\include
C:\ti\controlSUITE\device_support\f2806x\v141\F2806x_common\include



A.



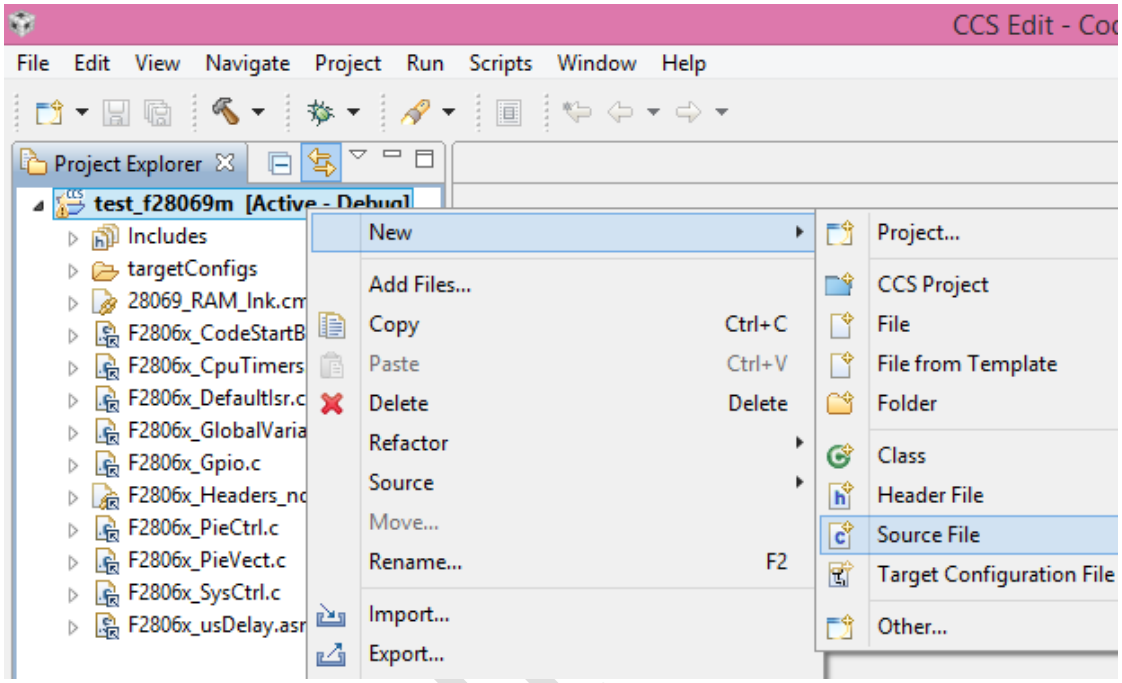
B.

Fig. 10 – A. – Fereastra pentru particularizare a proiectului (Properties for test_f28069m)
B. – Fereastra pentru definire a directoarelor de legătură

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pasul 5 – Crearea unui fișier sursă „C standard / C++” care să conțină codul programului:

Pentru a creea un nou fișier sursă, care va conține codul programului, în cadrul ferestrei de navigare în proiect, se va efectua comanda „click dreapta” asupra titlului proiectului activ în desfășurare „test_f28069m [Active – Debug]”, iar din meniu, se va alege opțiunea „New” → „Source File”. În fereastra nou deschisă, se vor păstra opțiunile implicite, iar în cadrul căsuței pentru numele fișierului (eng. Source file) se va trece „test_blink.c”;



A.

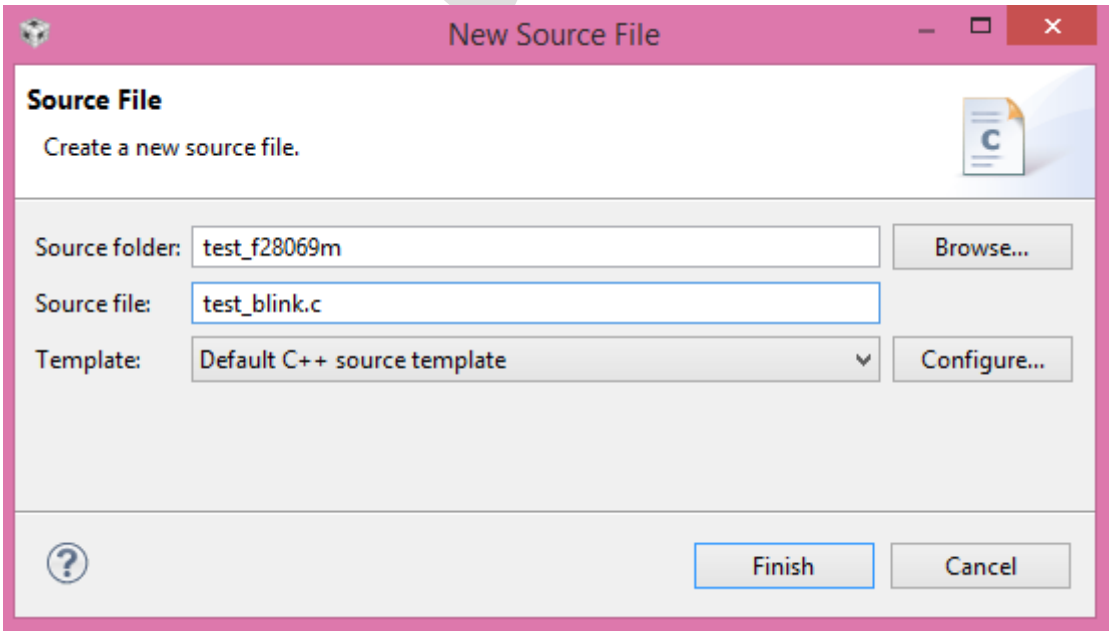


Fig. 11 – A. Crearea unui fișier C / C++ nou
B. Denumirea fișierului C / C++

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În cadrul fișierului C / C++ nou deschis, se vor introduce următoarele instrucțiuni:

```
#include "DSP28x_Project.h"

__interrupt void cpu_timer0_isr(void);

void main(void)
{
   InitSysCtrl(); // Initializare periferice + ceas
   DINT; // Dezactivarea intreruperilor generate de procesor
   InitPieCtrl(); // Initializarea registrilor de control PIE
   IER = 0x0000; // Dezactivarea intreruperilor generate de procesor si resetarea
indicatorilor
   IFR = 0x0000; // Dezactivarea intreruperilor generate de procesor si resetarea
indicatorilor
   InitPieVectTable(); // Initializare vector de intreruperi PIE
   EALLOW;  // Permiterea inscrierii registrilor protejati
   PieVectTable.TINT0 = &cpu_timer0_isr; // Redenumirea intreruperilor sub numele
ISR
   EDIS;    // Protejarea registrilor la inscriere
   InitCpuTimers();   // Initializare modul de temporizare (eng. Timer)
   ConfigCpuTimer(&CpuTimer0, 80, 500000); // Initializare modul de temporizare 80
Mhz 500 ms
   CpuTimer0Regs.TCR.all = 0x4001; // Protejare registii la inscriere TSS bit = 0
   EALLOW;
   GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0; // Initializare registru de control GPIO 0
   GpioCtrlRegs.GPADIR.bit.GPIO0 = 1; // Stabilire GPIO 0 ca si iesire digitala
   GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0; // Initializare registru de control GPIO 1
   GpioCtrlRegs.GPADIR.bit.GPIO1 = 1; // Stabilire GPIO 1 ca si iesire digitala
   EDIS;
   IER |= M_INT1; // Activare intrerupere procesor pentru modulul de temporizare
   PieCtrlRegs.PIEIER1.bit.INTx7 = 1; // Stabilirea ordinii in cadrul vectorului
de intreruperi
   EINT;   // Activarea intreruperilor la nivel global
   ERTM;   // Activarea intreruperilor pentru modul de functionare in timp real
   for(;;); // Bucla infinita
}


__interrupt void cpu_timer0_isr(void)
{
   CpuTimer0.InterruptCount++; //Numarare intreruperi
   GpioDataRegs.GPATOGGLE.bit.GPIO0 = 1; // Activare iesire digitala GPIO 0  la
500 ms
   GpioDataRegs.GPATOGGLE.bit.GPIO1 = 1; // Activare iesire digitala GPIO 0  la
500 ms
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Trimitere semnal de finalizare cu
succes a intreruperii
}
```

Pasul 6 – Generarea fișierului „.out” executabil:

În urma redactării codului de program, din bara de instrumente se va alege opțiunea „Build ‘Debug’ for project ‘test_f28069m’”. Mediul de programare Code Composer Studio va începe procesul de link – editare, compilare, și generare a fișierului „.out” executabil, care urmează a fi încărcat pe platforma F28069M LaunchPAD. Dacă nu apar erori în toate etapele descrise, atunci, în cadrul directoarelor de proiect se va genera fișierul executabil.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

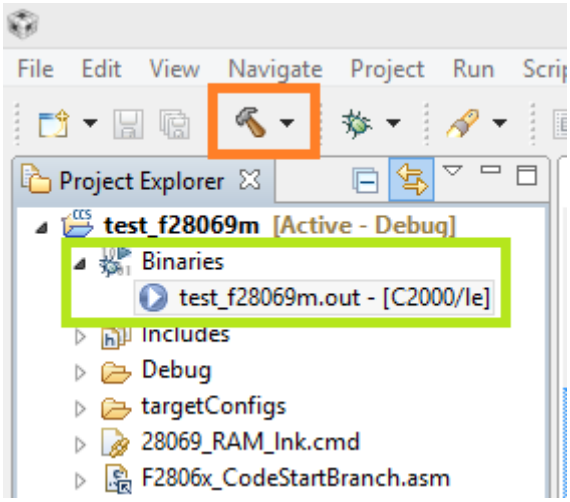


Fig. 12 – Opțiunea „Build 'Debug' for project" din bara de instrumente (marcaj portocaliu)
Fișierul binar sau executabil cu extensie „.out" generat în urma compilării (marcaj verde)

Pasul 7 – Încărcarea pe platforma F28069M LaunchPAD a fișierului executabil și rularea lui:

Tot în cadrul barei de instrumente se regăsește butonul „Debug" (depanare). Prin intermediul acestei funcții programul poate fi încărcat temporar în memoria RAM a platformei sau permanent în memoria FLASH. Încărcarea temporară a programului în memoria RAM a platformei, permite modificarea sau depanarea acestuia, chiar și în timpul execuției, dacă parametrii sunt adjustabili în timp real. Magistrala internă jTAG pentru diagnoză și depanare inter-conectează toate perifericele, procesorul și memoria platformei. Astfel, utilizând adaptorul TI XDS100v2 jTAG la USB, mediul Code Composer Studio va avea deci acces la toate datele vehiculate între periferice intern, fără a perturba execuția programului. Deci, toată activitatea internă a platformei poate fi monitorizată la nivel de regiștrii în cadrul CCS.

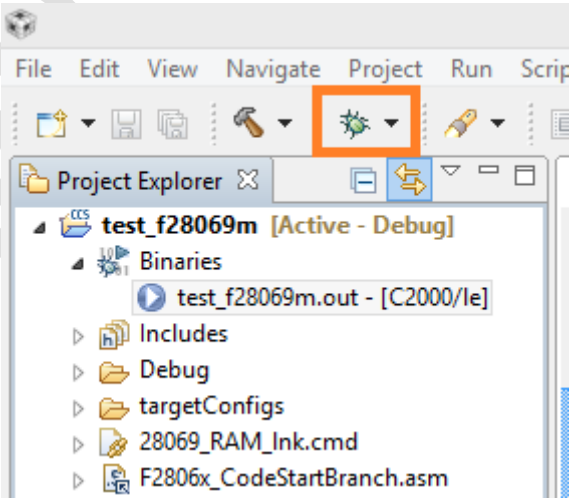


Fig. 13 – Opțiunea „Depanare" (eng. Debug) din bara de instrumente (marcaj portocaliu)

După alegerea opțiunii de „Depanare", mediul Code Composer Studio se va comuta în perspectiva de depanare, adică își va re-organiza sistemul de ferestre astfel încât să faciliteze

analiza codului scris în paralel cu instrucțiunile de asamblare și conținutul regiștrilor. În etapa aceasta, **este absolut necesară conectarea platformei de lucru la calculatorul gazdă!**

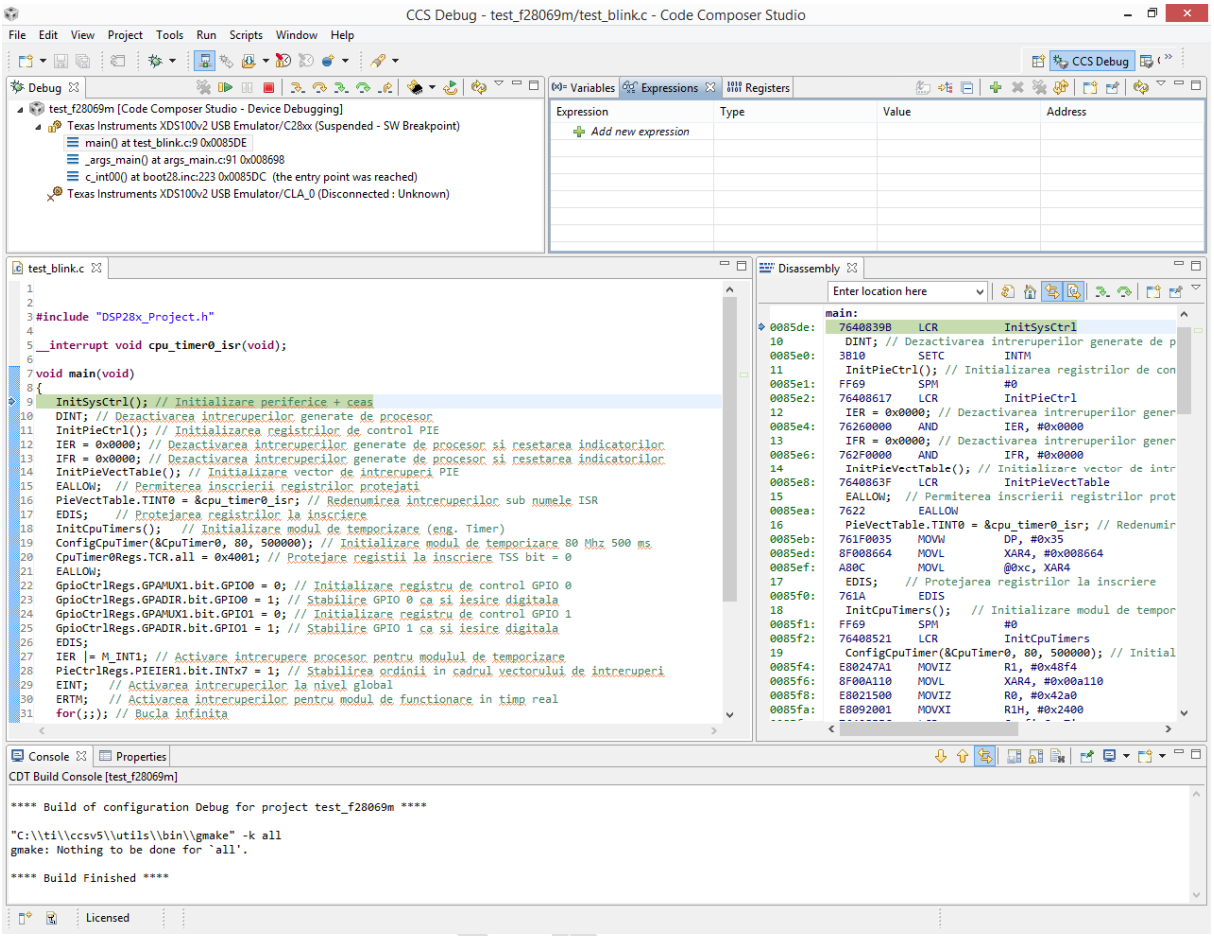


Fig. 14 – Mediul Code Composer Studio în perspectiva de depanare

Pentru a rula programul pe platforma F28069M din bara de instrumente a perspectivei de depanare, se va alege opțiunea „Reluare” (eng. Resume).

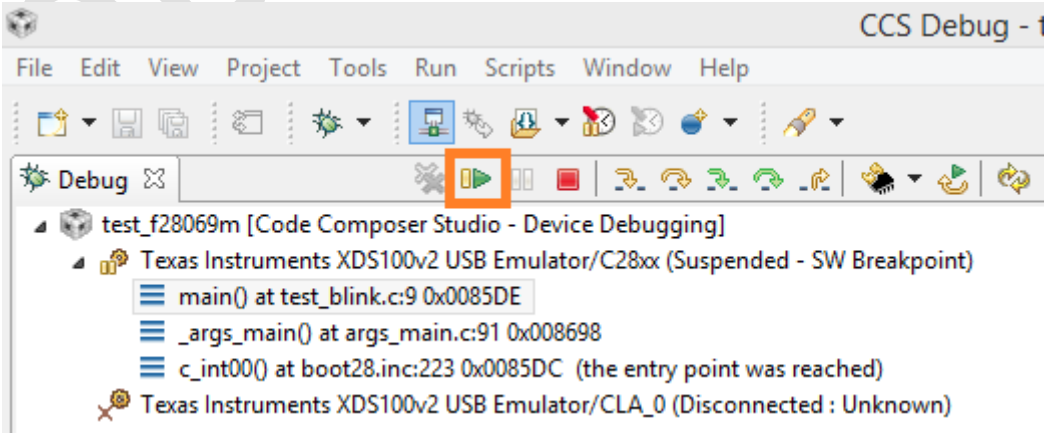


Fig. 15 – Bara de instrumente cu opțiunea Reluare (eng. Resume) (marcaj portocaliu)

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

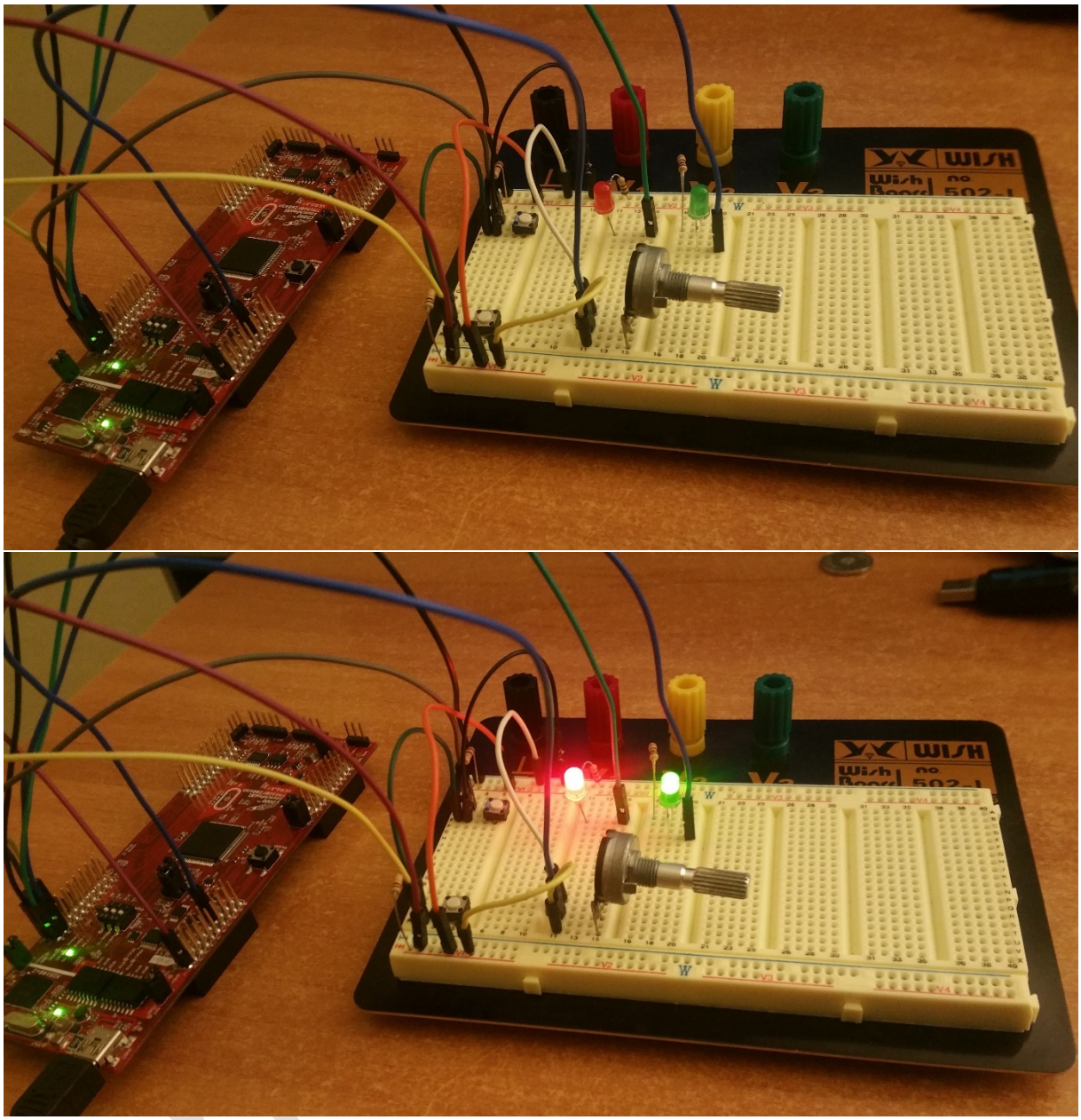


Fig. 16 – Rularea programului pe platforma C2000 - TMS320 - F28069M LaunchPAD

Rularea programului conceput constă în semnalizarea intermitentă a diodelor electroluminiscente în mod simultan.

Pasul 8 – Analiza rulării programului la nivelul platformei:

Mediul Code Composer Studio, permite monitorizarea sau înscrierea variabilelor în program în timp real prin accesul direct la anumite zone de memorie, cu ajutorul interfeței jTAG. Astfel, se va monitoriza numărul de întreruperi realizate cu succes la nivel de procesor, pentru a rula programul încărcat. În cadrul codului program a fost prevăzută o variabilă – contor de întreruperi (ex. CpuTimer0.InterruptCount++). Prin intermediul instrumentelor de analiză puse la dispoziție de către mediul Code Composer Studio, se va monitoriza în timp real valoarea acestui variabile, astfel:

- În partea dreaptă a programului dispus în perspectiva de depanare, există zona instrumentelor de analiză. În această zonă, pot fi declarate variabile, expresii, sau pot fi monitorizate conținuturile regiștrilor în timp real.
- Pentru a vizualiza variabila contor de întreruperi, în cadrul secțiunii „Expressions” se va declara numele variabile care se dorește a fi studiată, anume „CpuTimer0.InterruptCount”.
- Tot în cadrul zonei instrumentelor de analiză, se află și butonul „Continuous Refresh”. Se va activa această opțiune, pentru monitorizarea în timp real a variabilei declarate;
- Ultimul aspect, în bara generală de instrumente se va activa opțiunea „Enable Silicon Real-Time Mode” (execuția programului în timp real la nivel de circuit integrat);
- În final, pentru execuția programului se va acționa butonul „Resume”;

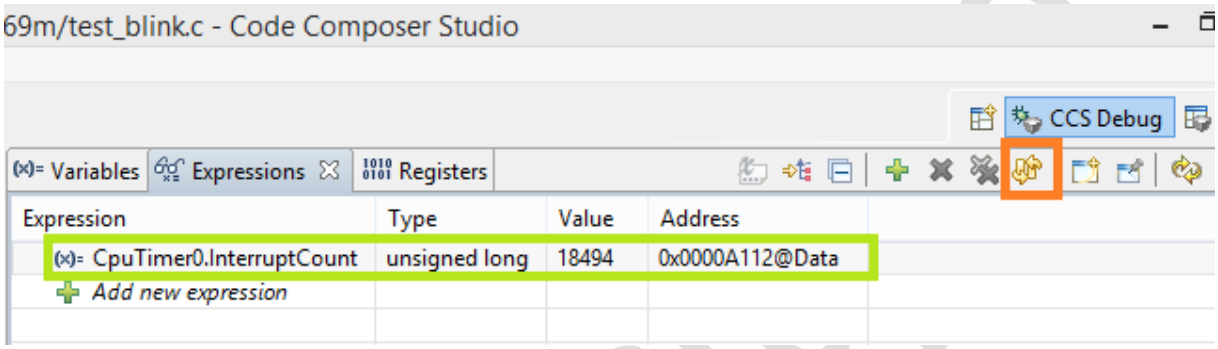


Fig. 17 – Zona instrumentelor de analiză a execuției – opțiunea „Continuous Refresh” (marcaj portocaliu), variabila „CpuTimer0.InterruptCount” (marcaj verde)

Fig. 18 – Bara generală de instrumente, opțiunea „Enable Silicon Real-Time Mode”

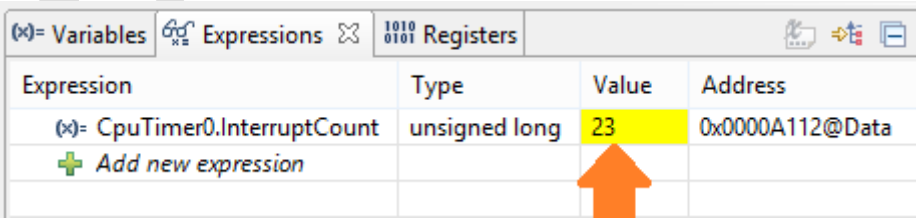


Fig. 19 – Actualizarea în timp real a valorii variabilei „CpuTimer0.InterruptCount”

Variabila „CpuTimer0.InterruptCount” se va incrementa în timp real la fiecare ciclu de întrerupere declanșat și realizat cu succes.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

B. Medode automate pentru generare a proiectului și a codului program:

2. Mediul de simulare și testare Matlab – Simulink și Code Composer Studio (versiunea 3.3):

- Reprezinta o soluție potrivită pentru cazul în care, procesorul digital de semnal, este utilizat într-o aplicație complexă pentru soluționarea unei probleme de inginerie. Această abordare, vizează modul de tratare sistematică a problemei de inginerie. Astfel, prin intermediul diagramei modelului Matlab – Simulink, în mod automat, se poate genera codul program, care stă la baza funcționării unui sistem implementat în Matlab - Simulink.

Scopul primordial, al cuplării celor două medii de simulare și de programare, este, de a putea rezolva problemele specifice domeniului ingineriei, pe baza unui sistem de calcul specializat, și dotat cu periferice specifice domeniului de destinație. Spre exemplu, stabilizarea tensiunii de la ieșire în cazul unui convertor coborâtor de curent continuu, necesită rapiditatea în ceea ce privește frecvența de calcul și soluționare, de asemenea este nevoie de un sistem de achiziție precum convertorul analog - digital, prin intermediul căruia să poată fi preluată o tensiune proporțională cu mărimea de măsură (ex. curent, tensiune, sau mărimi neelectrice). În aceeaș măsură, este nevoie de un generator de semnale, prin intermediul căruia să poată fi furnizată o comandă digitală, în funcție de logica de control implementată. Astfel pe baza unui sistem de calcul cu procesor digital de semnal, se pot construi unități centralizate pentru soluționarea buclelor de reglare digitală într-un sistem de natură analogică. Menționăm faptul că, pentru a realiza o aplicație care să permită interacțiunea bidirecțională în timp real, dintre mediul Matlab – Simulink și aplicația fizică (eng. hardware) dezvoltată pe baza platformei F28069M LaunchPAD, sunt necesare următoarele programe:

- Code Composer Studio 3.3;
- ControlSUITE;
- Matlab R2015b + Simulink + Embedded Coder + C2000 Toolbox;

În vederea generării automate a codului program și a proiectului, pe baza diagramei realizate în mediul Matlab – Simulink se va proceda astfel:

Pasul 1 – Configurarea platformei F28069M în mediul Code Composer Studio 3.3:

Mediul Code Composer Studio 3.3 are două puncte de configurare anume:

- Setup Code Coposer Studio 3.3;
- Code Composer Studio 3.3 IDE;

Prin intermediul componentei „Setup Code Composer Studio 3.3”, se oferă posibilitatea utilizatorului, de a particulariza proprietățile funcționale ale platformei de dezvoltare utilizată în cadrul mediului de dezvoltare „Code Composer Studio 3.3”. Aceste proprietăți funcționale, constau în: modelul adaptorului USB – jTAG, modelul procesorului și pachetul de instrucțiuni utilizate în vederea stabilirii comunicării dintre platforma de dezvoltare și calculatorul gazdă. În cazul platformei C2000 - F28069M se procedează astfel:

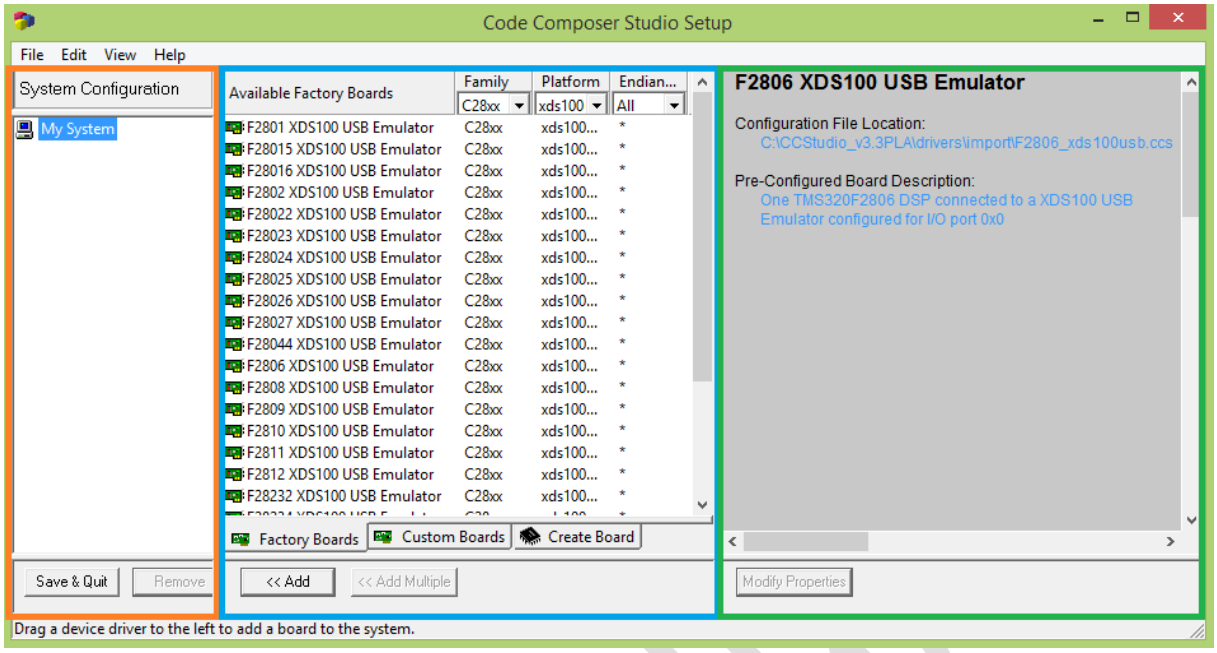
În cadrul componentei „Setup Code Coposer Studio 3.3”, există trei zone, anume:

- zona pentru afișare a configurației platformă – calculator gazdă (f. 20 - marcaj portocaliu);
- lista tuturor platformelor și adaptoarelor jTAG disponibile în baza de date (marcaj albastru);
- zona pentru descriere a platformei selectate (marcaj verde);

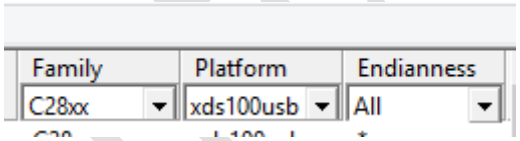
În partea de sus a listei, se vor alege parametrii: Family: „C28xx”, Platform: „xds100usb”, Endianness: „All”, Din cadrul listei se va alege opțiunea „F2806 XDS 100

ee.utcluj.ro

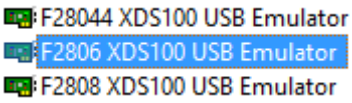
Emulator”. În final, se va proceda prin apăsarea butoanelor „Add” și „Save and Quit”. **Menționăm faptul că platforma trebuie să fie conectată la calculator în această etapă!**



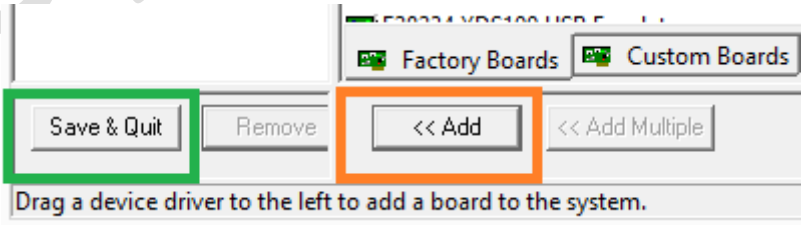
A.



B.



C.



D.

Fig. 20 – A. - Componenta Setup Code Coposer Studio 3.3, B. - Parametrii de căutare în listă, C. - Opțiunea F2806 XDS100, D. – Butoanele „Add” și „Save & Quit”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În urma parcurgerii etapelor menționate, mediul Code Composer Studio 3.3, va fi lansat în execuție în mod automat. Pentru a testa conexiunea dintre platforma de dezvoltare și calculatorul gazdă, din meniul „Debug” (depanare), se va alege opțiunea „Connect” (conectare). În cazul în care conexiunea este realizată cu succes, va fi afișată fereastra dezasamblorului de cod „Disassembly”.

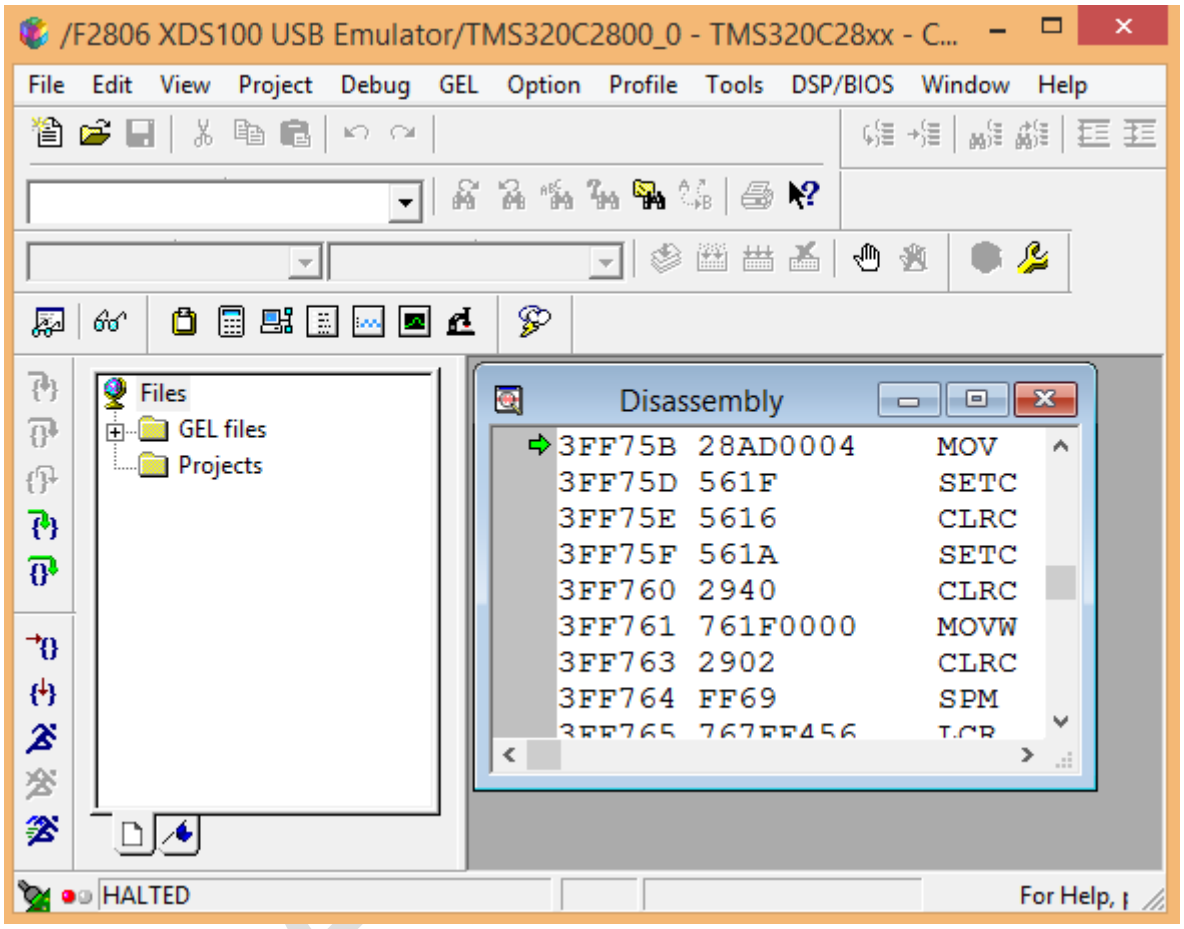


Fig. 21 – Mediul Code Composer Studio 3.3 conectat la platforma de dezvoltare

Dacă etapele amintite au fost parcurse cu succes, se va proceda la închiderea mediului Code Composer Studio 3.3, urmând ca acesta să fie inițializat prin intermediul consolei de comandă din cadrul mediului Matlab.

Pasul 2 – Conectarea mediului de simulare Matlab – Simulink la Code Composer Studio 3.3:
- Pentru a realiza acestă etapă, un prim pas imperios necesar, va fi lansarea în execuție CU DREPTURI ADMINISTRATIVE, mediul Matlab – Simulink versiunea 2015b. (Pentru a executa cu drepturi administrative o aplicați în sistemul de operare Windows, este suficient dacă se efectuează comanda click dreapta pe pictograma programului, iar din meniul deschis se alege opțiunea „Run as administrator”). În urma lansării în execuție a mediului Matlab, va apărea fereastra de bază a aplicației, în care este conținută și consola de comandă (fig. 22 - marcaj verde). Înainte de a furniza orice comandă în consola de comandă Matlab, se va stabili un director pentru spațiul de lucru, utilizând bara și fereastra de navigație (fig. 22 - marcaj roșu).

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

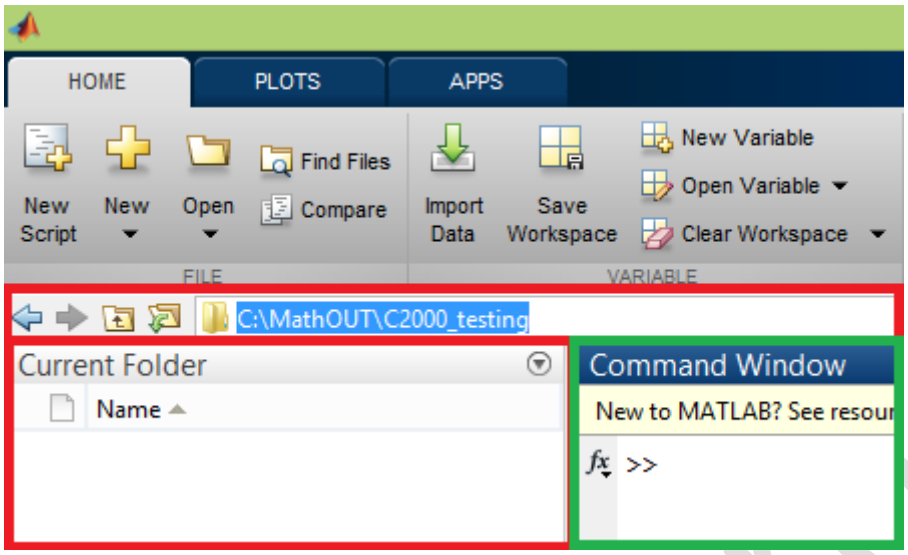


Fig. 22 – Mediul Matlab

- În urma specificării directorului de lucru, în consola de comandă se vor introduce comenzile:

```
cc = ticcs
cc.visible(1)
```

După introducerea primei comenzi, în consola de comandă vor apărea câteva informații despre platforma de dezvoltare definită în Setup Code Composer Studio, iar după introducerea comenzii următoare, se va deschide mediul Code Composer Studio 3.3. Astfel, între cele două programe, a fost sabilită legătura de comunicare. O ultimă comandă va fi:

```
simulink
```

```
>> cc = ticcs

TICCS Object:
  Processor type    : TMS320C28xx
  Processor name    : TMS320C2800_0
  Running?          : No
  Board number      : 0
  Processor number  : 0
  Default timeout   : 10.00 secs

  RTDX channels     : 0

>> cc.visible(1)
>> simulink
>>
```

Fig. 23 – Consola de comandă a mediului Matlab

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pasul 3 – Configurarea modelului pentru generare automată de cod:

- Pentru a genera în mod automat codul programului care urmează a fi încărcat în memoria platformei de dezvoltare F28069M, se va creea un model Simulink nou. Modelul nou creeat, se va salva sub numele test_f28069m, în directorul de lucru stabilit. După parcurgerea acestor etape, se va deschide fereastra de lucru pentru conceperea modelului.

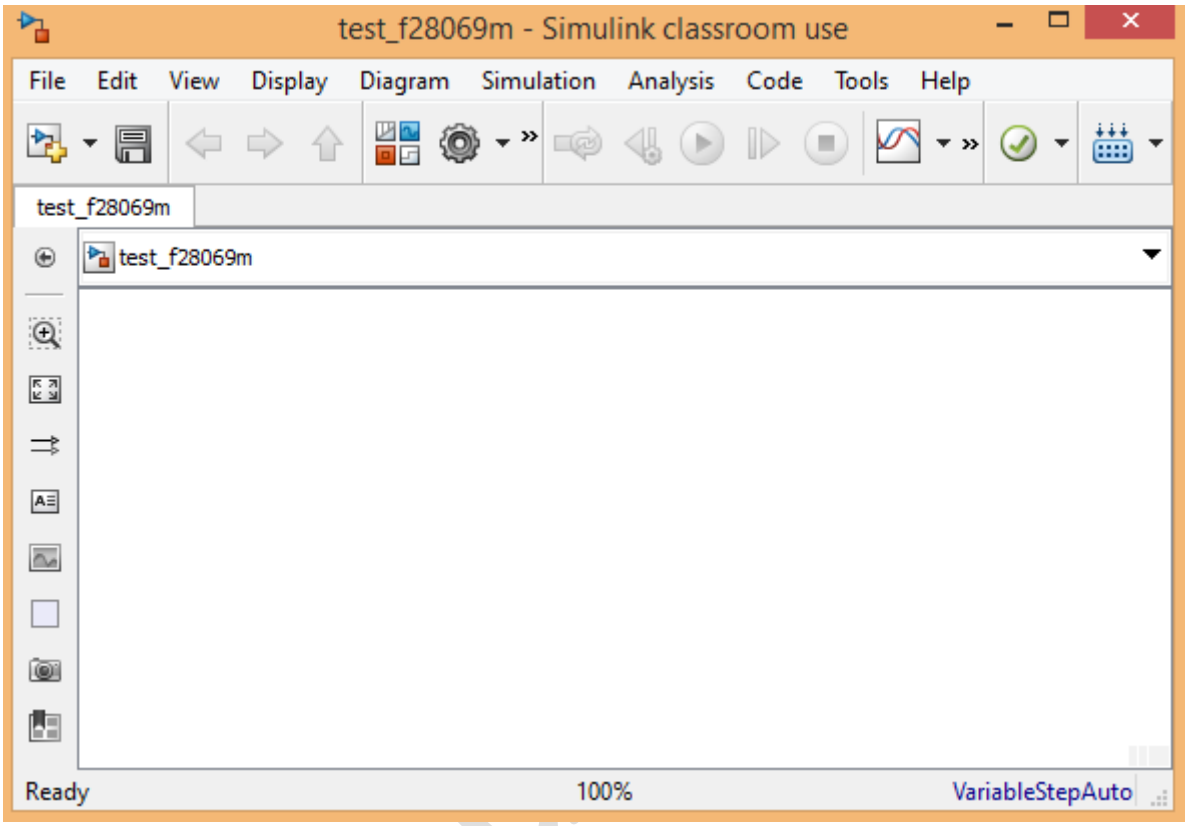


Fig. 24 – Fereastra de lucru pentru conceperea modelului

- În vederea conceperii unui model pe baza căruia să poată fi generat în mod automat codul sursă al programului care va rula pe platforma de dezvoltare, este necesară parametrizarea inițială a modelului nou creat. Astfel, din meniul „Simulation” se va alege opțiunea „Model Configuration Parameters. În cadrul ferestrei nou deschise se vor avea în vedere următoarele categorii de setări „Solver” și „Code Generation”. Comutarea categoriilor se realizează din lista situată în partea stângă a ferestrei.

În categoria „Solver”, se va opta pentru următorii parametrii:
- Start time: 0.0;
- Stop time: 100;
- Type: Fixed-step;
- Solver: discrete (no continuous states);

În cadrul grupului „Additional options”:
- Fixed-step size (fundamental sample time): 1e-4;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

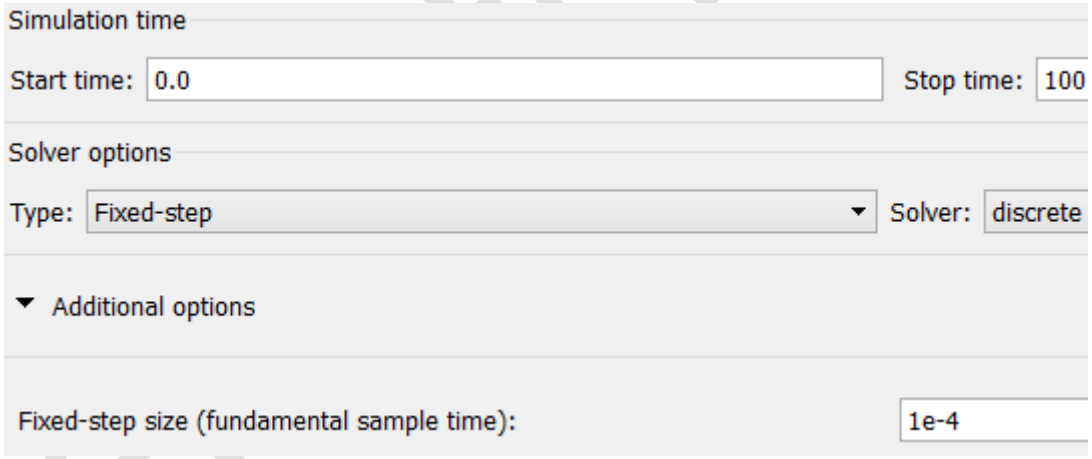
În categoria „Code Generation” se vor stabili următorii parametrii:
- System target file: (prin intermediul butonului „Browse”): idelink_ert.tlc;

În cadrul sub-categoriei „Coder Target” (apartenetă categoriei principale „Code Generation”), există două pagini pentru parametrizare, anume: „Tool Chain Automation” și „Target Hardware Resources”. În cadrul paginii „Tool Chain Automation” se vor impune parametrii:
- Build format: Project;
- Build action: Build;
- Configuration: Debug;

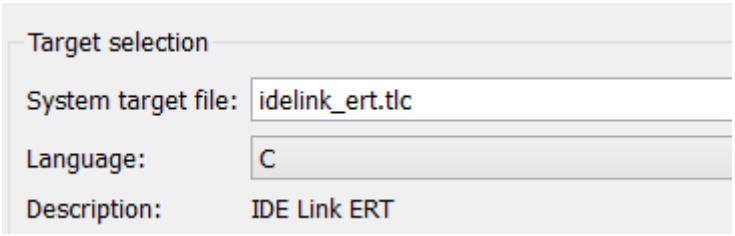
În cadrul paginii „Target Hardware Resources” se vor impune parametrii:
- IDE / Tool Chain: Texas Instruments Code Composer Studio (este vorba de versiunea 3.3);

În cadrul grupului „Board Properties”:
- Board: C2000 Custom;
- Processor: F28069_cpu;

În cadrul grupului „IDE Support”, se va apăsa butonul „Get from IDE” pentru a obține datele necesare despre adaptorul USB – jTAG și tipul procesorului digital de semnal;
Finalizarea parametrizării se va realiza prin apăsarea butonului „Apply” apoi „Ok”;

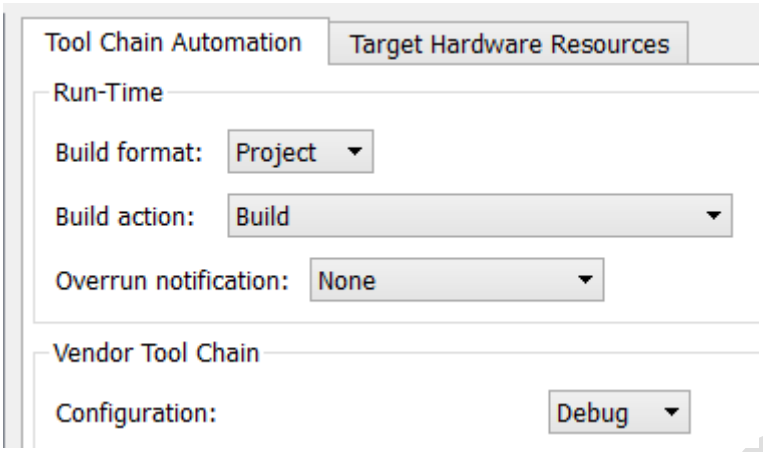


A.

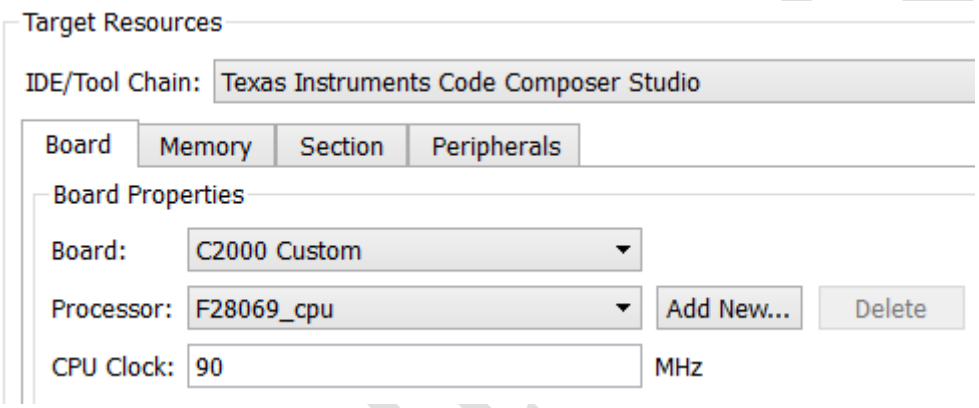


B.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro



C.



D.

E.

Fig. 25 – A - Categoria „Solver”, B - categoria „Code Generation”,
C - Pagina „Tool Chain Automation”, D - pagina „Target Hardware Resources”
E - grupul „IDE Support”

Pasul 4 - Conceperea modelului Simulink pentru programul intern al platformei:

- În vederea conceperii unui model Simulink, compatibil cu procesorul platformei F28069M, pe lângă instrumentele standard puse la dispoziție de mediul Matlab – Simulink, se vor utiliza și instrumentele specifice platformei din cadrul paletei de instrumente „Embedded Coder Support Package for Texas Instruments C2000 Processors”. Paletele de instrumente se pot accesa prin intermediul modulului de gestiune a bibleotecilor (Simulink Library Browser).

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

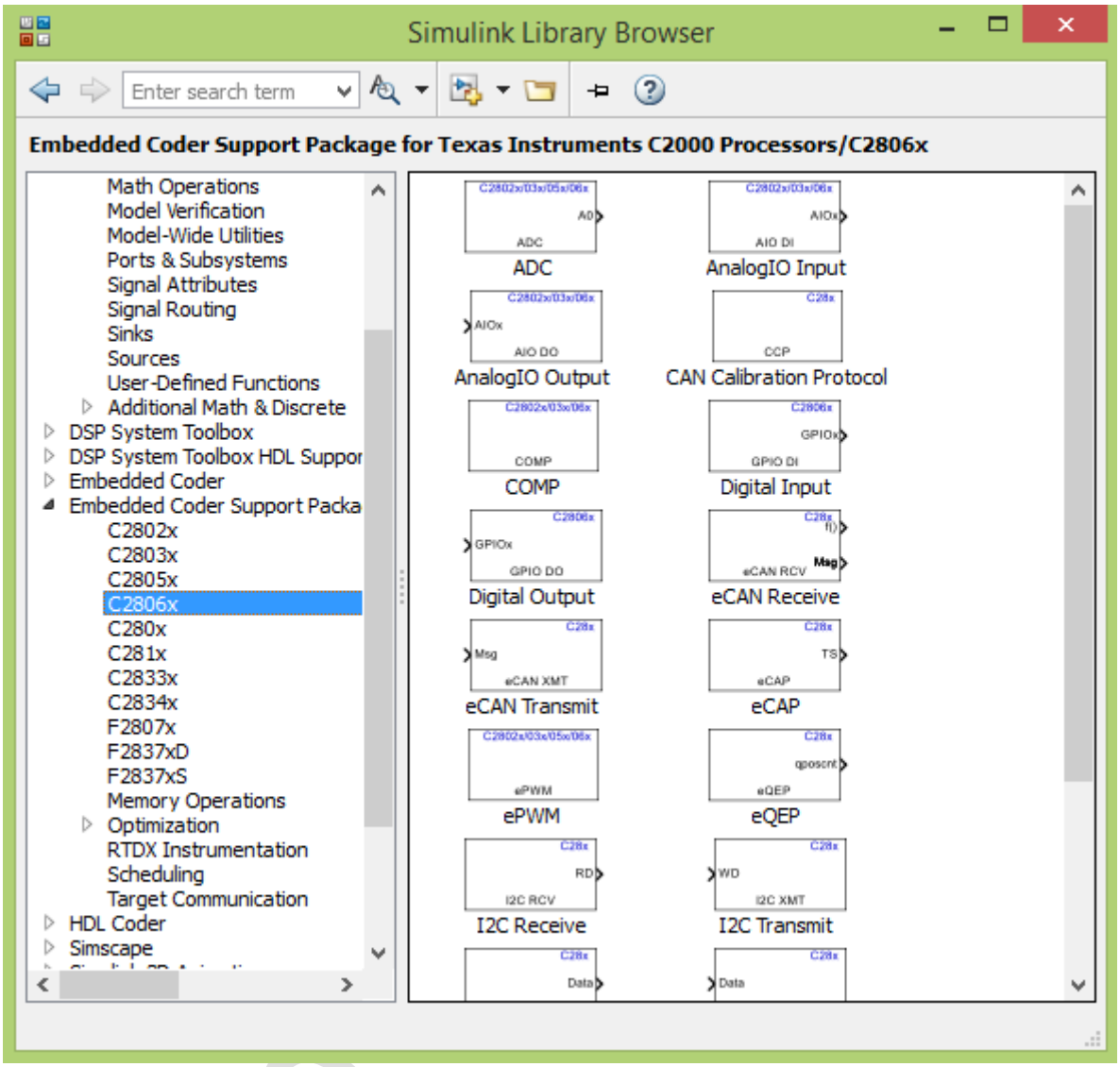


Fig. 26 – Paleta de instrumente specifice platformei C2000 – F28069M

- Cu ajutorul acestor instrumente se va concepe următorul model:

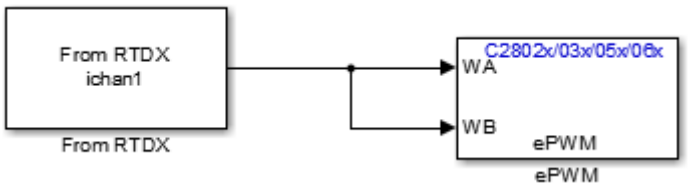


Fig. 27 – Modelul programului ce urmează a fi încărcat în memoria platformei F28069M

- Rolul programului simbolizat prin modelul conceput, este de a prelua valoarea numerică a factorului de umplere (0% - 100%) de la calculatorul gazdă prin intermediul interfeței jTAG cu ajutorul protocolului Real - Time Data eXchange (RTDX). Blocul „From RTDX” se găsește în categoria „Embedded Coder Support Package for Texas Instruments C2000 Processors” și sub-

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

categoria „RTDX Instrumentation”. Blocul „ePWM” se găsește în sub-categoria „C2806x”. În vederea adăugării conectorilor „WA” și „WB” blocului „ePWM”, vor fi necesare următoarele serii de parametrizări (fereastra de parametrizare se obține prin comanda dublu click):

În categoria „General” , pe lângă valorile implicite, se vor impune următorii parametrii:
- Timer period units: Seconds;
- Specify timer period via: Specify via dialog;
- Timer period: 0.0001;
- Counting mode: Up;

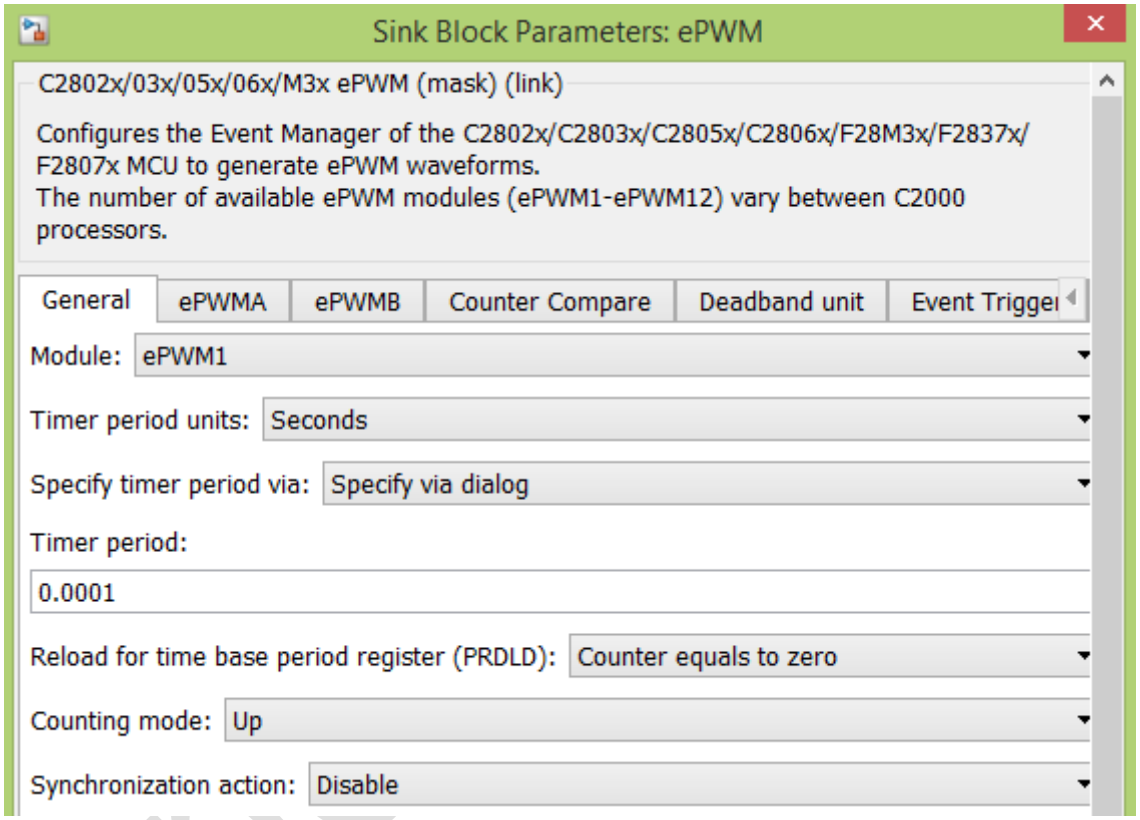


Fig. 28 – Ferestra de parametrizarea a blocului „ePWM” – categoria „General”

În categoria „Counter Compare”:
- CMPA units: Percentages;
- Specify CMPA via: Input Port;
- CMPA initial value: 0;
- Reload for compare A Register (SHDWAMODE): Counter equals to zero;
- CMPB units: Percentages;
- Specify CMPB via: Input port;
- CMPB initial value: 0;
- Reload for compare B Register (SHDWBMODE): Counter equals to zero;

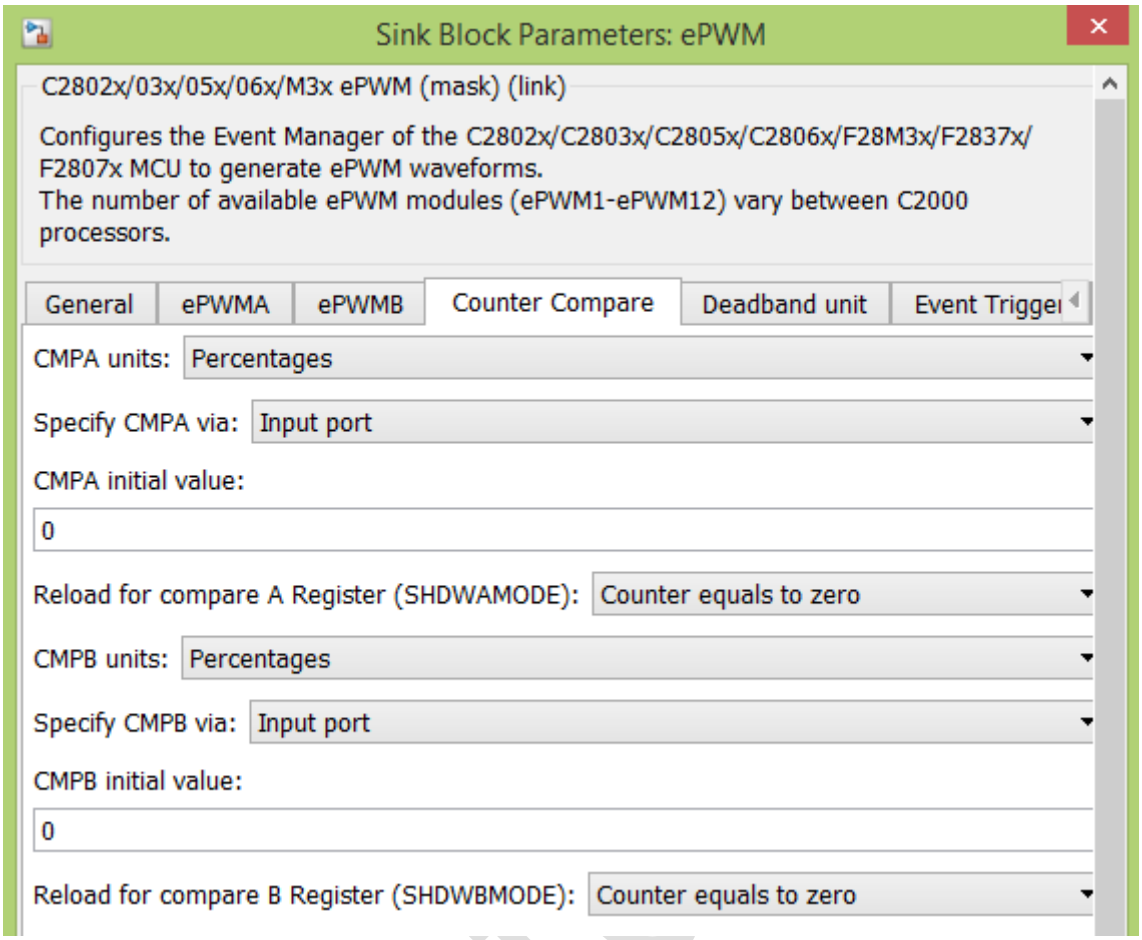


Fig. 29 - Ferestra de parametrizarea a blocului „ePWM” – categoria „Counter Compare”

Finalizarea procesului de parametrizare a blocului „ePWM” se va realiza prin apăsarea butoanelor „Apply” opoi „Ok”.

Pentru blocul „From RTDX” se vor introduce următorii parametrii:
- Channel name: ichan1;
- Se va debifa căsuța: „Enable blocking mode”;
- Initial condition: 0;
- Sample Time: 1e-4;
- Output dimensions: [1 32]:
- Se va debifa căsuța: „Frame-based”;
- Data type: int32;
- Se va bifa căsuța: „Enable RTDX channel on start-up”;

Finalizarea procesului de parametrizare a blocului „From RTDX” se va realiza prin apăsarea butoanelor „Apply” opoi „Ok”.

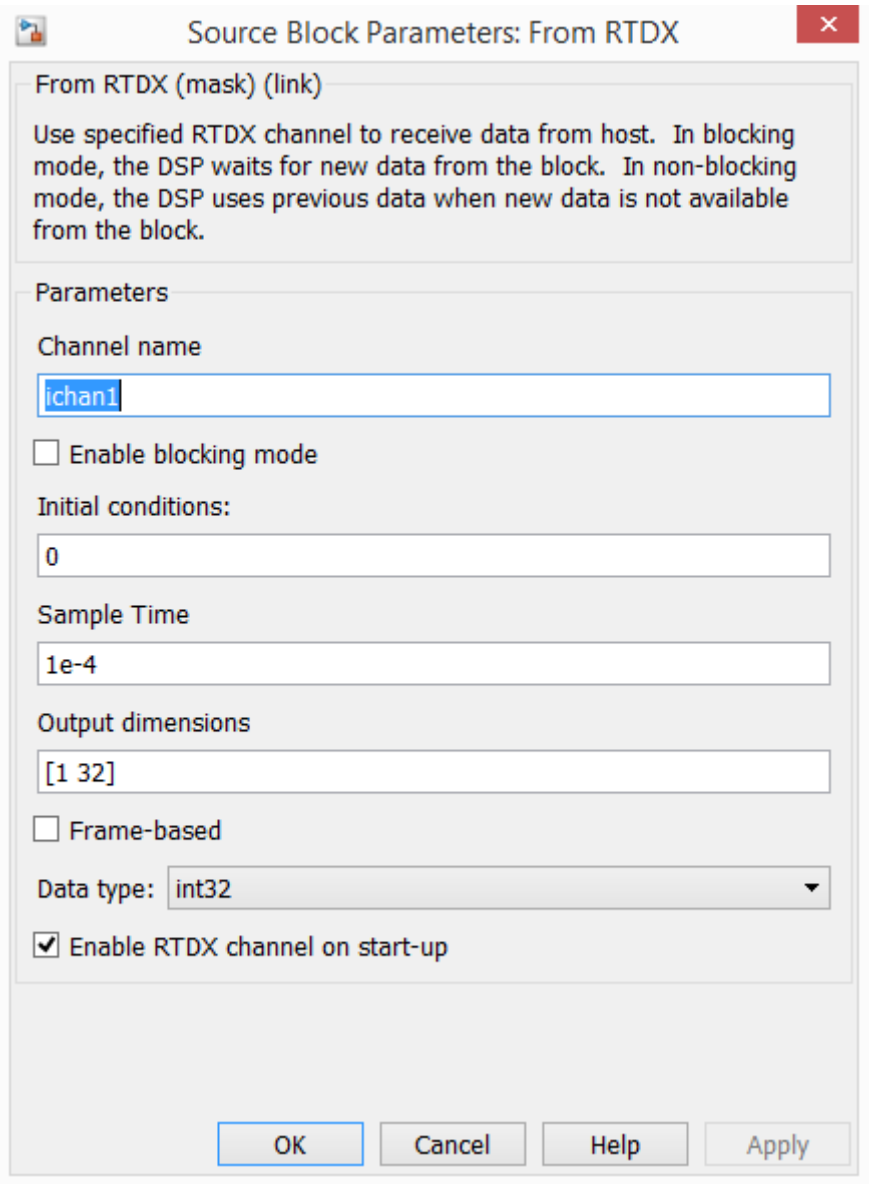


Fig. 30 - Ferestra de parametrizarea a blocului „From RTDX”

Pentru a genera în mod automat codul programului, în mediul Simulink în meniul „Code” → „C / C++ code” se va alege opțiunea „Build Model”

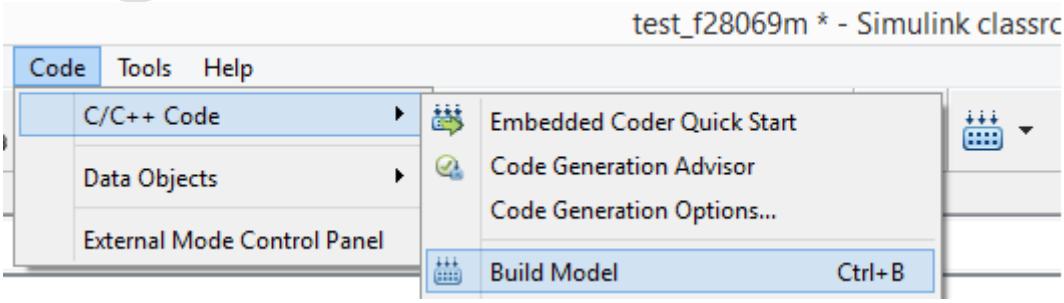


Fig. 31 – Generarea automată a codului program

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Datorită legăturii create inițial între mediul Matlab – Simulink și mediul Code Composer Studio 3.3, prin intermediul extensiei funcționale (eng. plug – in) „idelink_ert.tlc”, generatorul de cod „Embedded Coder” al mediului Matlab - Simulink are posibilitatea să compileze modelul creeat sub formă de diagramă în cod C / C++ compatibil cu mediul Code Composer Studio (conform parametrizărilor care au fost impuse – a se vedea pașii anteriori!).

După finalizarea procesului de compilare al modelului Simulink, fișierele C / C++ rezultate, vor fi preluate în mod automat în mediul Code Composer Studio 3.3. Odată ce fișierele necesare au fost preluate, se va inițializa în mod automat un proiect nou pe baza acestor fișiere (etapa de pre – link – editare). Odată finalizată, etapa de creeare a proiectului, mediul Code Composer Studio, va iniția etapa de compilare și generarea fișierului executabil, „.out” care ulterior va fi încărcat în memoria platformei F28069M la cerere.

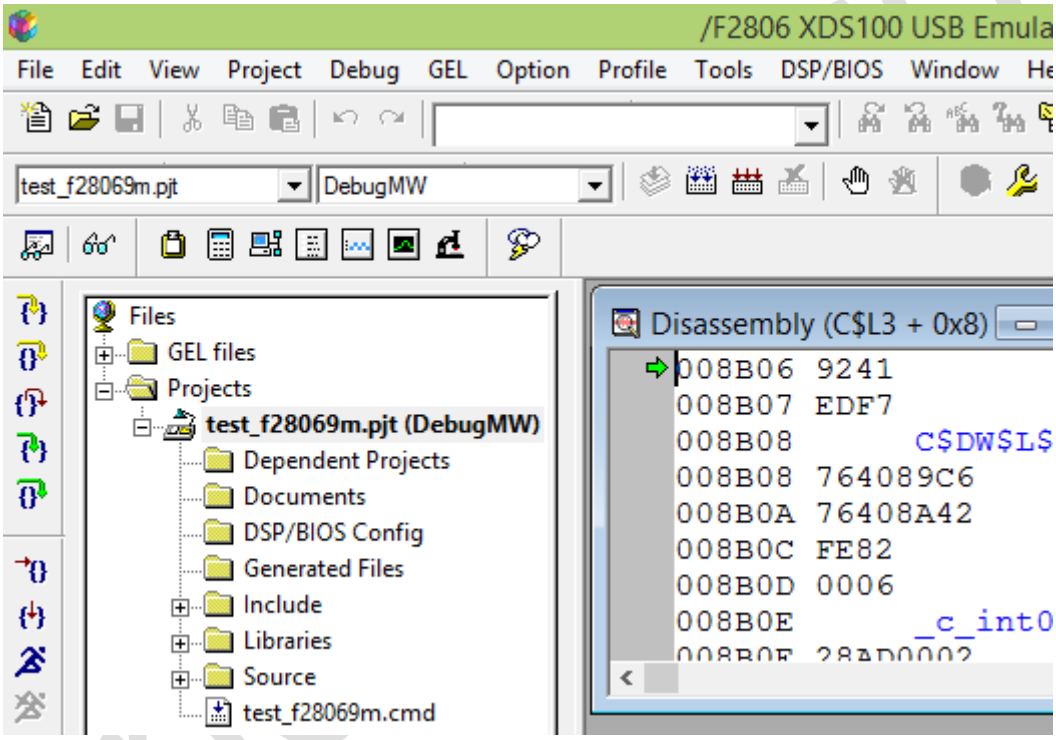


Fig. 32 – Proiectul generat în mediul Code Composer Studio 3.3

În această etapă, la nivelul platformei de dezvoltare F28069M LaunchPAD, programul este pregătit, gata de a fi încărcat și lansat în execuție. Singurul impediment, constă în faptul că, din punct de vedere al execuției, utilizatorul nu are nici o cale de comunicare cu programul care rulează în interior, decât canalul de comunicație jTAG – Real Time Data eXchange inițiat prin intermediul programului conceput în model (adică, „ichan1”).

Pentru a putea furniza valoarea numerică a factorului de umplere pentru generatorul de semnal modulat în lățime (ePWM), este necesar să se creeze un alt model care să ruleze pe calculatorul gazdă. Astfel, prin intermediul adaptorului jTAG – USB - XDS100v2 încorporat pe placă (eng. ON – Board) se va putea realiza un canal de comunicație între cele două echipamente, anume „ichan1”. În acest sens, la nivelul calculatorului gazdă, se va realiza următorul model Simulink:

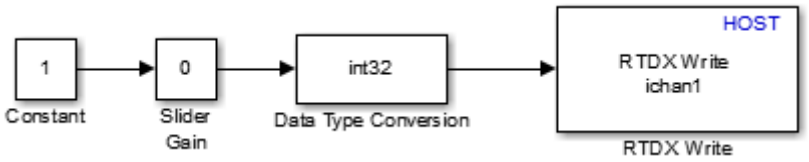


Fig. 33 – Model de control, necesar calculatorului gazdă pentru a furniza înspre platforma F28069M valoarea numerică a factorului de umplere

Blocurile componente ale modelului pot fi regăsite în prima categorie de bază „Simulink”:
- Blocul „Constant” se găsește în sub-categoria „Sources”;
- Blocul „Slider Gain” se găsește în sub-categoria „Math Operations”;
- Blocul „Data Type Conversion” se găsește în sub-categoria „Signal Attributes”;
Blocul „RTDX Write” (HOST) se obține prin introducerea în consolă a comenzii:

```
rtdxsimlib
```

Odată introdusă, această comandă va genera o bibleotecă Simulink, (aparent ascunsă) prin intermediul căreia, mediul Simulink rulând pe calculatorul gazdă și având legătura stabilită cu mediul Code Composer Studio 3.3, va avea posibilitatea de a comunica în timp real prin interfața jTAG cu aplicația care rulează în memoria platformei F28069M.

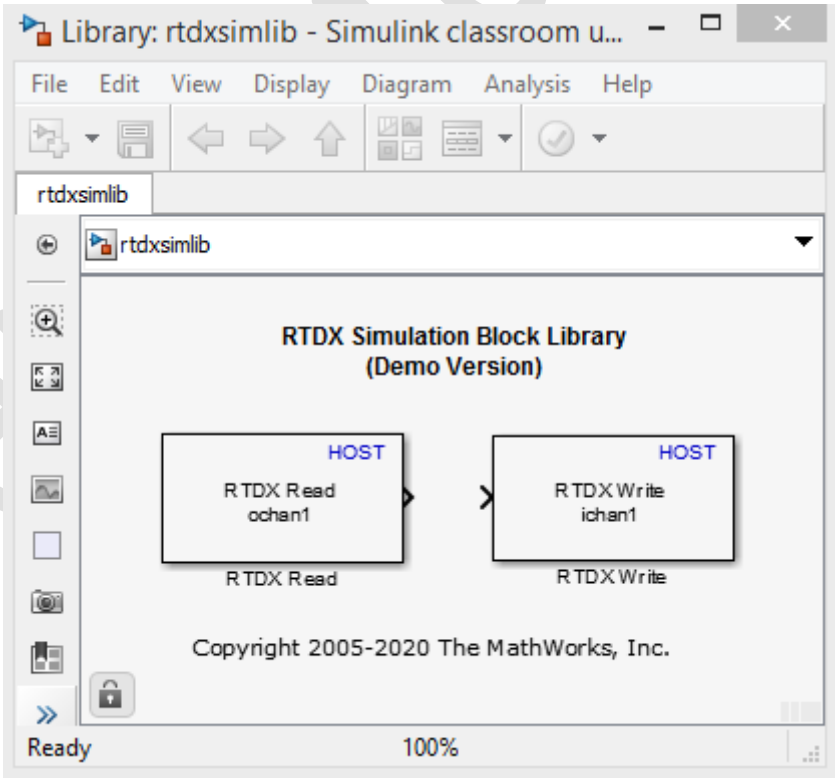


Fig. 34 – Bibleoteca Real Time Data eXchange – HOST

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În urma realizării modelului de control indicat precedent, se va proceda la stabilirea legăturii dintre proiectul CCS_3.3 actual și acest model. În meniul „Simulation” se va alege opțiunea „Model Configuration Parameters”. Se observă faptul că, în urma introducerii blocurilor de comunicare RTDX ale calculatorului gazdă, în fereastra „Model Configuration Parameters” apare o nouă categorie de parametrii, anume „Host-Target Communication”.
În această categorie, se va alege opțiunea „Select” (buton aflat în zona „Target Information”). O casetă de dialog va confirma faptul că a fost identificată platforma de dezvoltare.

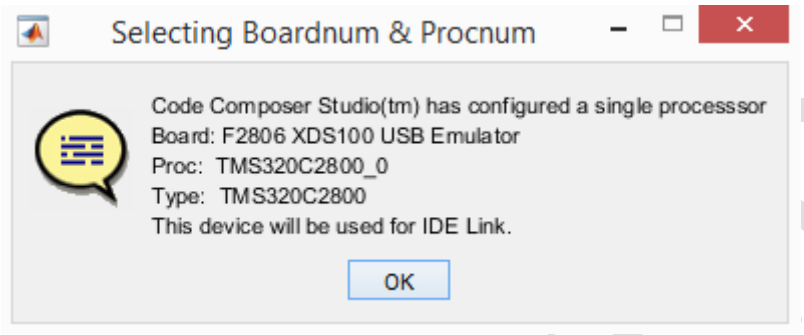


Fig. 35 – Identificarea platformei de dezvoltare

În zona „Target Application” se vor introduce căile de acces înspre fișierul proiectului generat, și fișierul executabil generat. Locația acestor fișiere, de obicei, este în directorul de lucru al modelului, care a fost stabilit la început. În cazul de față „C:\MathOUT\C200_testing”. Sub-directorul „test_f28069m_ticcs” a fost creeat de către mediul Code Composer Studio 3.3, odată cu etapa de compilare a proiectului. În acest director se vor regăsi atât fișierul de proiect (în cazul de față „test_f28069m.pjt”) cât și fișierul executabil („test_f28069m.out).

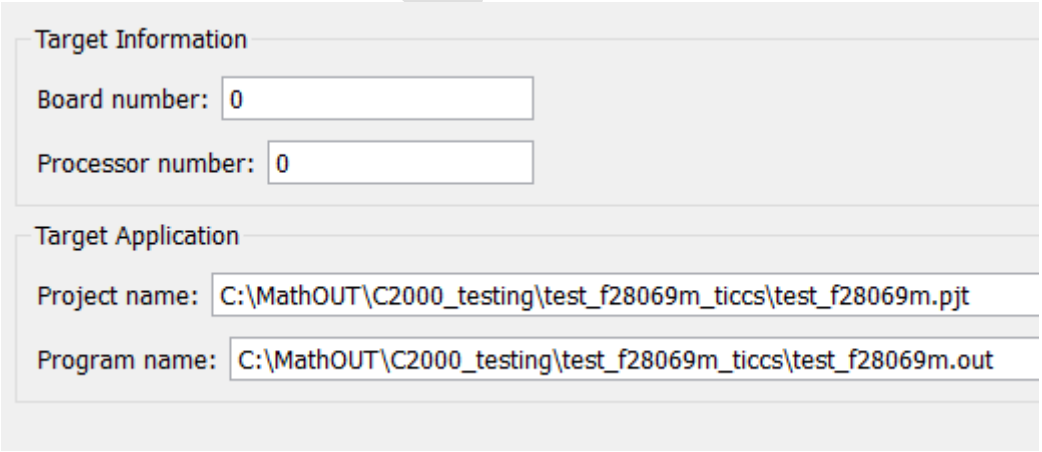


Fig. 36 – Localizarea fișierelor necesare pentru a inița comunicarea dintre calculatorul gazdă și platforma de dezvoltare F28069M

Finalizarea procesului de parametrizare al modelului se realizează prin apăsarea butoanelor „Apply”, apoi „Ok”.

Blocul „Constant” are valoarea „1” predefinită, deci nu se vor mai face alte parametrizări la acest bloc. Blocul „Slider Gain” reprezintă un factor de amplificare (înmulțire) variabil. Prin intermediul cursorului de care dispune acest bloc, utilizatorul, va avea posibilitatea să adjusteze în timp real factorul de umplere al generatorului de semnal programabil, apelat în programul care rulează pe platforma F28069M. Parametrii care pot fi introduși în cadrul acestui bloc, sunt extremele (superior „100” și inferior „0”). Acest bloc, necesită la intrarea sa, o constantă cu valoarea unitară, pentru a-i păstra scala de multiplicare constantă, crescătoare 1 la 1.

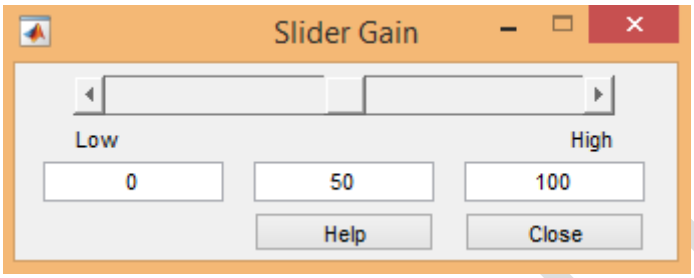


Fig. 37 – Factor de amplificare adjustabil prin intermediul cursorului (eng. Slider Gain)

Pentru că, factorul de umplere este exprimat în valori întregi de la 0 la 100, va fi nevoie să se realizeze conversia rezultatului în număr întreg cu reprezentare pe 32 de biți. Acest lucru poate fi realizat prin intermediul blocului „Data Type Conversion”. Parametrul care trebuie modificat în cadrul acestui bloc, se află în zona „Parameters”, anume, „Output data type”.

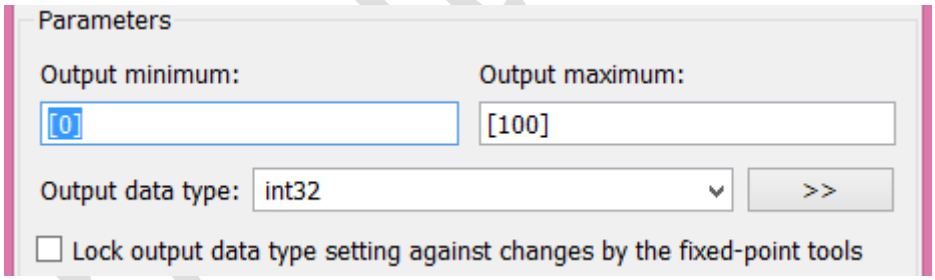


Fig. 38 – Conversia tipului de date la număr întreg cu reprezentare pe 32 de biți

În final prin intermediul blocului „RTDX Write” (HOST side), se va stabili numele canalului de comunicare prin interfața jTAG (același cu cel al platformei) anume „ichan1”.

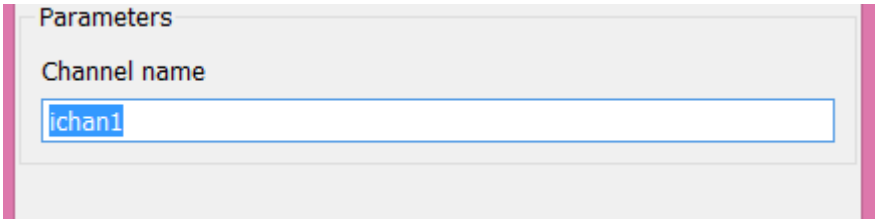


Fig. 39 – Stabilirea canalului de comunicare Real – Time Data eXchange la nivelul calculatorului gazdă

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

La finalul acestor etape, practic, au fost realizați toți pașii necesari pentru a implementa o aplicație de control în timp real pe baza mediului Matlab – Simulink, Code Composer Studio, și pe baza platformei de dezvoltare cu DSP Texas Instruments F28069M LaunchPAD. Pentru a lansa în execuție aplicația la nivelul procesorului digital de semnal și la nivelul calculatorului gazdă, în cadrul modelului aferent calculatorului gazdă, în bara de instrumente se va stabili timpul de rulare al aplicație 100 de secunde, după care se va acționa butonul „Run” (rulare):

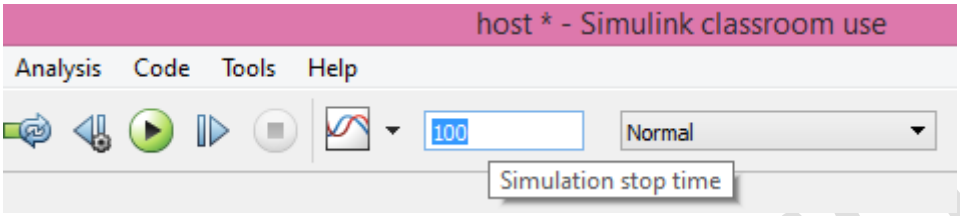


Fig. 40 – Bara de instrumente pentru controlul execuției modelului

În momentul apăsării butonului „Run” (rulare), mediul Code Compose Studio, va încărca aplicația executabilă în memoria platformei și va iniția canalul de comunicație RTDX.

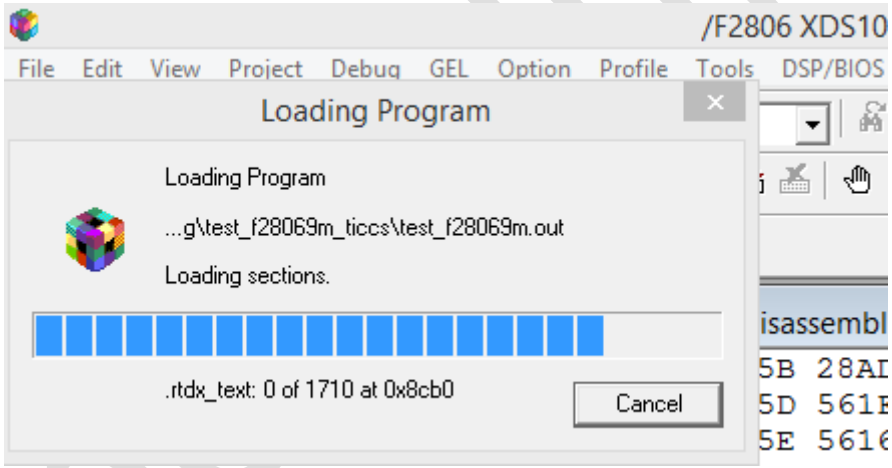
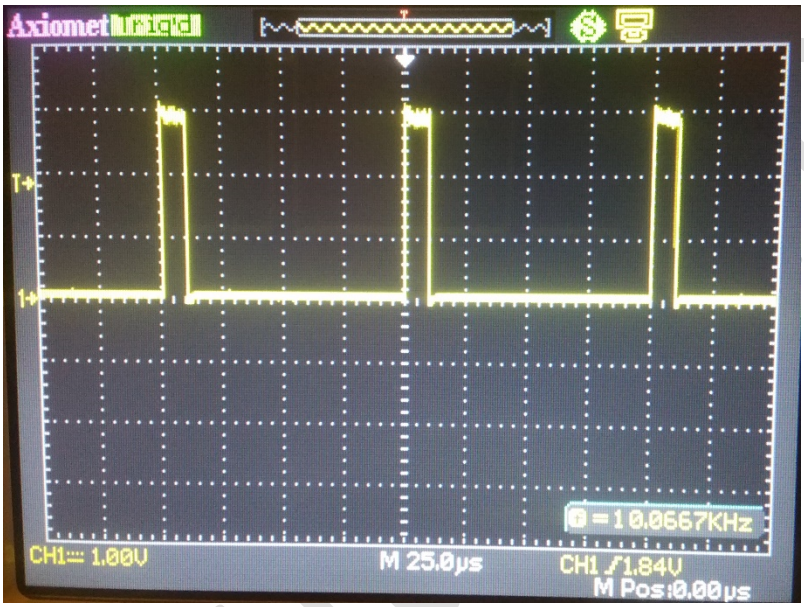


Fig. 41 – Încărcarea fișierului executabil „test_f28069m.out” în memoria platformei de dezvoltare F28069M LaunchPAD

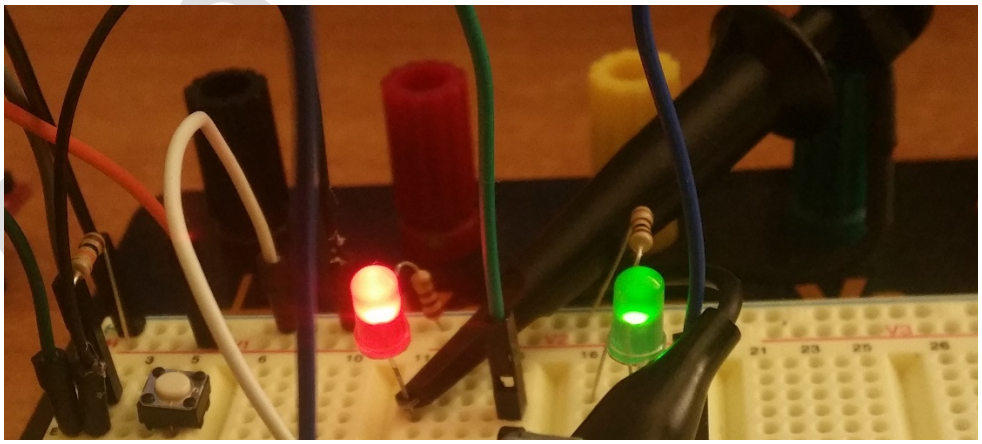
Prin intermediul cursorului „Slider Gain”, se poate adjusta în timp real, (în timp ce rulează simularea), factorul de umplere al semnalului modulat în lățime. Acest lucru, poate fi vizibil, în cazul în care la ieșirile digitale ale platformei de dezvoltare sunt atașate diode electroluminiscente sau un osciloscop. Efectul produs de variația factorului de umplere al unui semnal modulat în lățime aplicat ca și o tensiune asupra diodelor electroluminiscente, este cel de a produce variația intensității luminoase. Reprezentat pe osciloscop, acest semnal va avea o formă dreptunghiulară cu lățime variabilă.

Astfel se vor testa două cazuri, anume 10% și 80% factor de umplere, în care se vor constata efectele apărute:

A.



B.

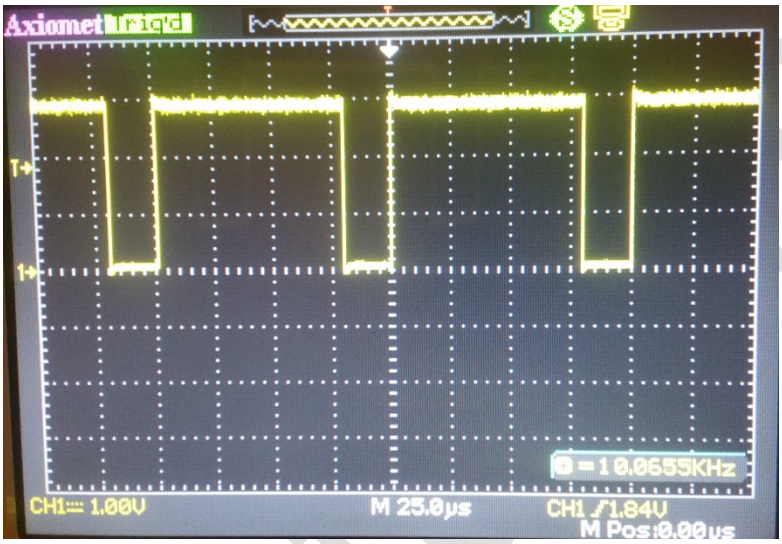


C.

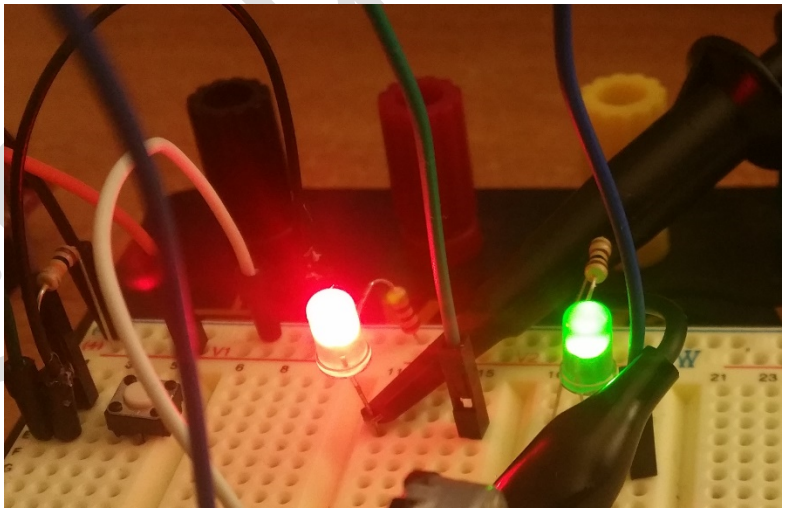
Fig. 42 – A. Adjustarea factorului de umplere la valoarea 10% prin intermediul cursorului
B. Oscilografierea semnalului modulat în lățime
C. Efectul semnalului asupra diodelor elecroluminiscente atașate la ieșiriele digitale

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

A.



B.



C.

Fig. 43 – A. Adjustarea factorului de umplere la valoarea 80% prin intermediul cursorului
B. Oscilografierea semnalului modulat în lățime
C. Efectul semnalului asupra diodelor elecroluminiscente atașate la ieșiriele digitale

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Codul programului generat automat:

```
/*
 * File: test_f28069m_main.c
 *
 * Code generated for Simulink model 'test_f28069m'.
 *
 * Model version                  : 1.7
 * Simulink Coder version         : 8.9 (R2015b) 13-Aug-2015
 * C/C++ source code generated on : Mon Apr 13 18:53:15 2020
 *
 * Target selection: idelink_ert.tlc
 * Embedded hardware selection: Texas Instruments->C2000
 * Code generation objectives: Unspecified
 * Validation result: Not run
 *
 */

#include "test_f28069m.h"
#include "rtwtypes.h"
#include "test_f28069m_private.h"
#include "c2000_main.h"
#include "F2806x_Device.h"
#include "F2806x_Examples.h"
#include <stdlib.h>
#include <stdio.h>

void init_board(void);
void enable_interrupts(void);
void config_schedulerTimer(void);
void disable_interrupts(void);
volatile int IsrOverrun = 0;
static boolean_T OverrunFlag = 0;

/* Function: rt_OneStep ---------------------------------------
 *
 * Abstract:
 *      Perform one step of the model.  Single-tasking implementation.
 */
void rt_OneStep(void)
{
  /* Check for overrun. Protect OverrunFlag against
   * preemption.
   */
```

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

```
  if (OverrunFlag++) {
    IsrOverrun = 1;
    OverrunFlag--;
    return;
  }

  asm(" SETC INTM");
  PieCtrlRegs.PIEIER1.all |= (1 << 6);
  asm(" CLRC INTM");
  test_f28069m_step();

  /* Get model outputs here */
  asm(" SETC INTM");
  PieCtrlRegs.PIEIER1.all &= ~(1 << 6);
  asm(" RPT #5 || NOP");
  IFR &= 0xFFFE;
  PieCtrlRegs.PIEACK.all = 0x1;
  asm(" CLRC INTM");
  OverrunFlag--;
}

/* Function: main -------------------------------------------
 *
 * Abstract:
 *      Entry point into the code.
 */
void main(void)
{
  volatile boolean_T noErr;
  init_board();
  rtmSetErrorStatus(test_f28069m_M, 0);
  test_f28069m_initialize();
  config_schedulerTimer();
  noErr =
    rtmGetErrorStatus(test_f28069m_M) == (NULL);
  enable_interrupts();
  while (noErr ) {
    noErr =
      rtmGetErrorStatus(test_f28069m_M) == (NULL);
  }

  /* Disable rt_OneStep() here */

  /* Terminate model */
```

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

```
 test_f28069m_terminate();
 disable_interrupts();
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

3. Mediul de simulare și testare SolidThinking Embed / Altair Embed / VisSim:

- Reprezintă soluția completă la întreg lanțul de simulare, testare a funcționalității în timp real a programului conceput, apoi inscripționarea permanentă a acestuia în memoria platformei.

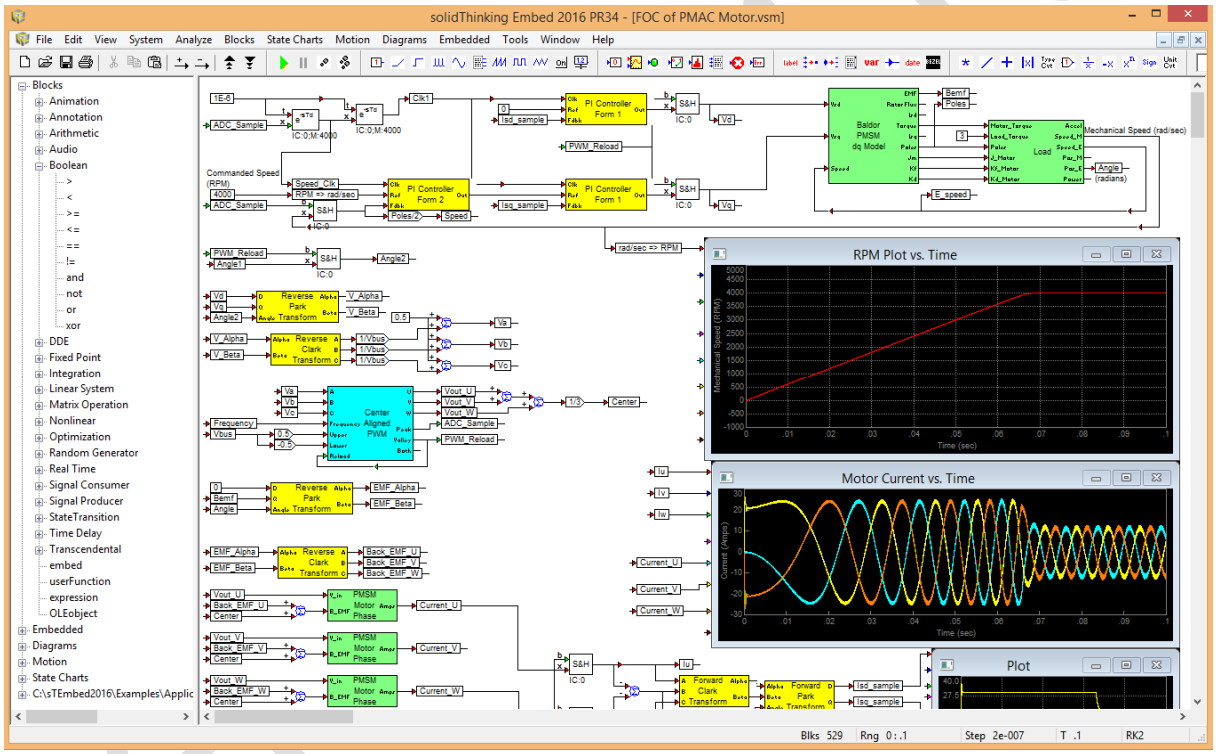


Fig. 44 - Mediul de simulare și testare SolidThinking Embed / Altair Embed / VisSim

În vederea realizării a unui program simplu, pentru platforma F28069M LaunchPAD, în cadrul mediului Altair Embed, se va proceda astfel:

Pasul 1 – Crearea unui model de simulare în timp real:

- În cadrul mediului Altair Embed, se vor închide oricare alte modele deschise, apoi, din meniul „File” se va alege opțiunea „New”. În urma efectuării operațiilor indicate, se va salva modelul nou creeat sub numele spre exemplu „test_f28069m”.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

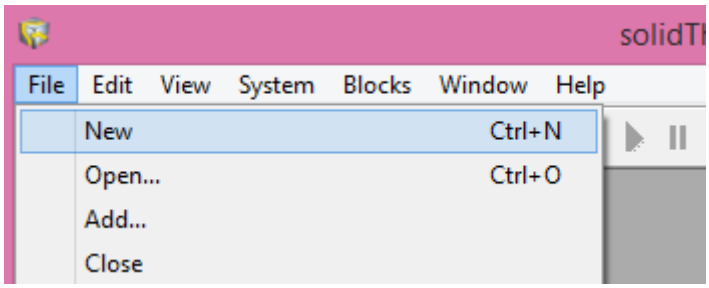


Fig. 45 – Crearea unui model nou

Pasul 2 – Parametrizarea modelului de simulare:

- Din meniul „System” se va alege opțiunea „System Properties...”:

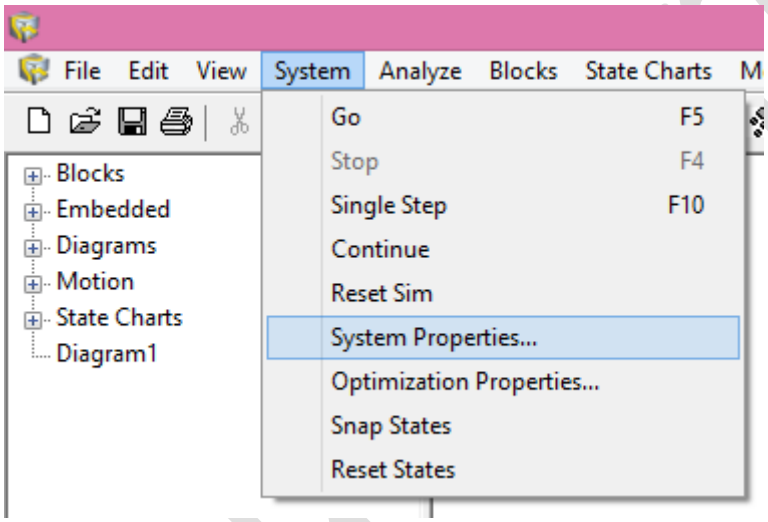


Fig. 46 – Parametrizarea modelului prin intermediul opțiunii „System Properties...”

În fereastra nou deschisă, se vor introduce următorii parametrii:

- Start (sec.): 0;
- Time Step: 1e-4; → Seconds;
- End (sec.): 100;
- Se va bifa opțiunea „Run in Real Time”;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

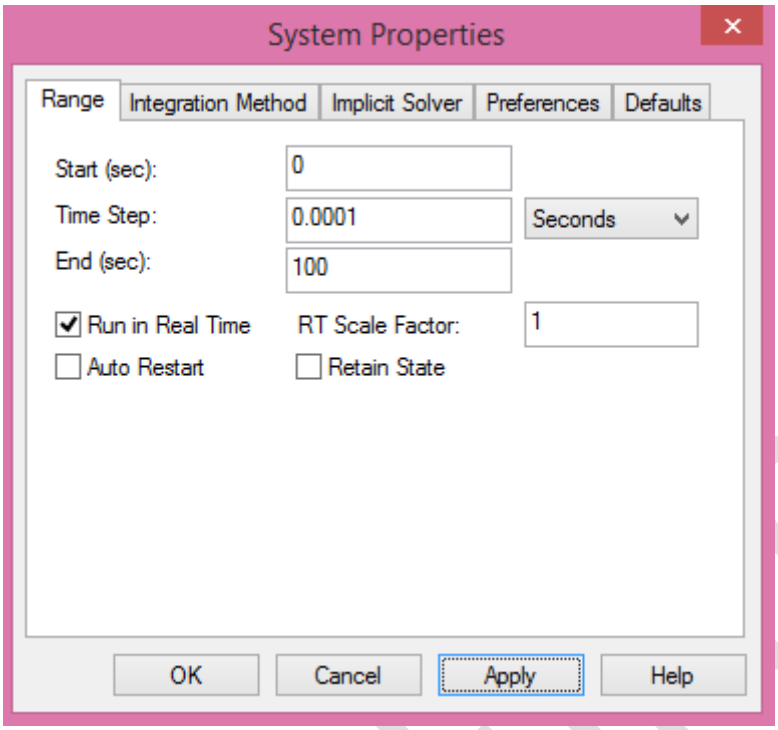


Fig. 47 – Fereastra „System Properties” pentru parametrizarea modelului

Procedura se va finaliza prin apăsarea butonului „Apply” apoi „Ok”.

Pasul 3 – Introducerea parametrilor platformei de dezvoltare în model:

- În vederea realizării acestui pas, din meniul „Embedded” → „Piccolo” se va selecta opțiunea „F28x Config...”. Rezultatul acestei opțiuni va fi afișarea unei ferestre de parametrizare.

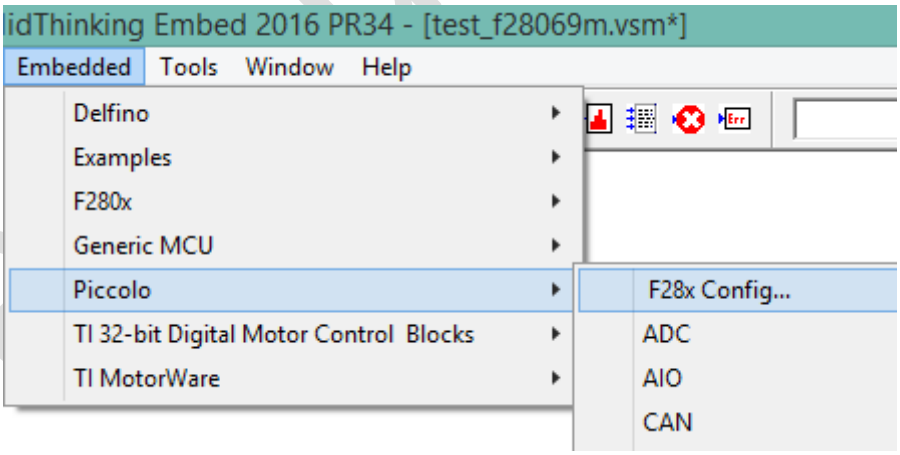


Fig. 48 – Alegerea opțiunii „F28x Config...”

În cadrul ferestrei de parametrizare a platformei de lucru se vor selecta următorii parametrii:

- CPU: F28069M;

- Opțiunea „Enable Interactive Peripheral Mode” va rămâne debifată;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- CPU Speed (MHz): 90;
- Clock Source: „Internal Oscillator 1”;
- Multiple of Crystal Freq: 9x;
- HSPCLK: 1/SYSCLK → 90 MHz;
- LSPCLK: 4/SYSCLK → 22.5 MHz;
- JTAG connection: TI XDS100v2 USB;
- Control Clk Src: 32 bit timer 0;
- EPWM Interrupt Event: CTR = 0;
- Control Clk Prescale: 1;
- Ctrl Clk Count Mode: Down;

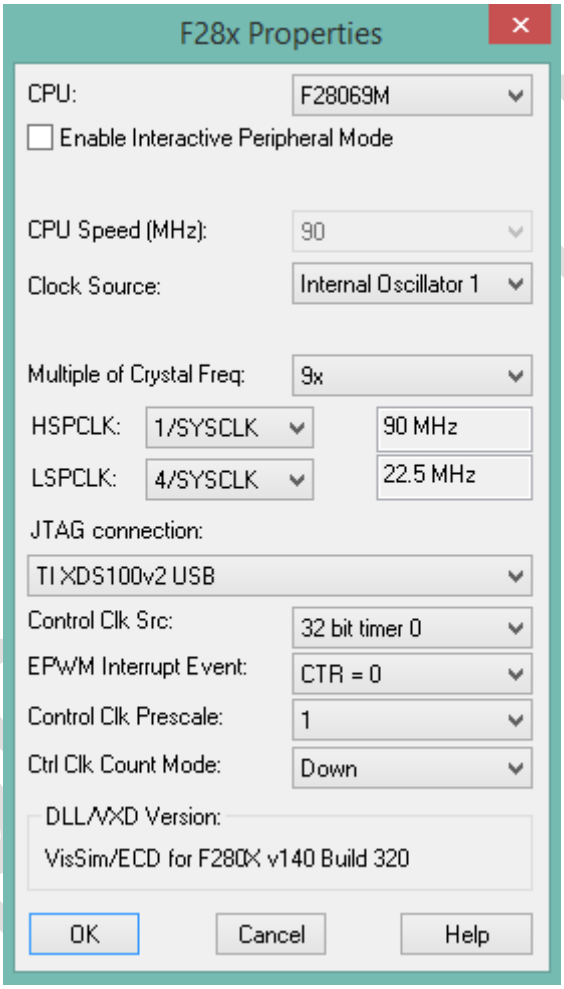


Fig. 49 – Fereastra „F28x Properties” pentru parametrizare a platformei

În urma tuturor parametrizărilor se va încheia procedura prin apăsarea butonului „Ok”, iar blocul rezultant, va fi plasat în spațiul de lucru gol (alb) al modelului.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pasul 4 – Introducerea blocurilor specifice perifericelor platformei de dezvoltare:

- În vederea adăugării unui bloc pentru controlul ieșirilor digitale de uz general (eng. GPIO) ale platformei cu DSP - F28069M, se va proceda astfel: Din meniul „Embedded” → „Piccolo” → „GPIO” → „GPIO Output for F280x”. Blocul rezultant va fi plasat în spațiul de lucru al modelului, după care, prin comanda „dublu click” asupra blocului introdus, se va deschide o fereastră de parametrizare. În categoria „Channel” se va introduce valoarea „0” (GPIO0). Se va finaliza parametrizarea blocului prin apăsarea butonului „Ok”. Se va proceda în mod similar și pentru a adăuga un al doilea bloc în vederea controlării ieșirii digitale „GPIO1”.

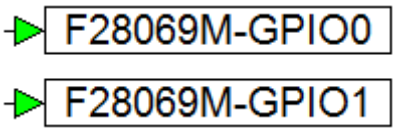


Fig. 50 – Blocurile rezultante pentru controlul ieșirilor digitale

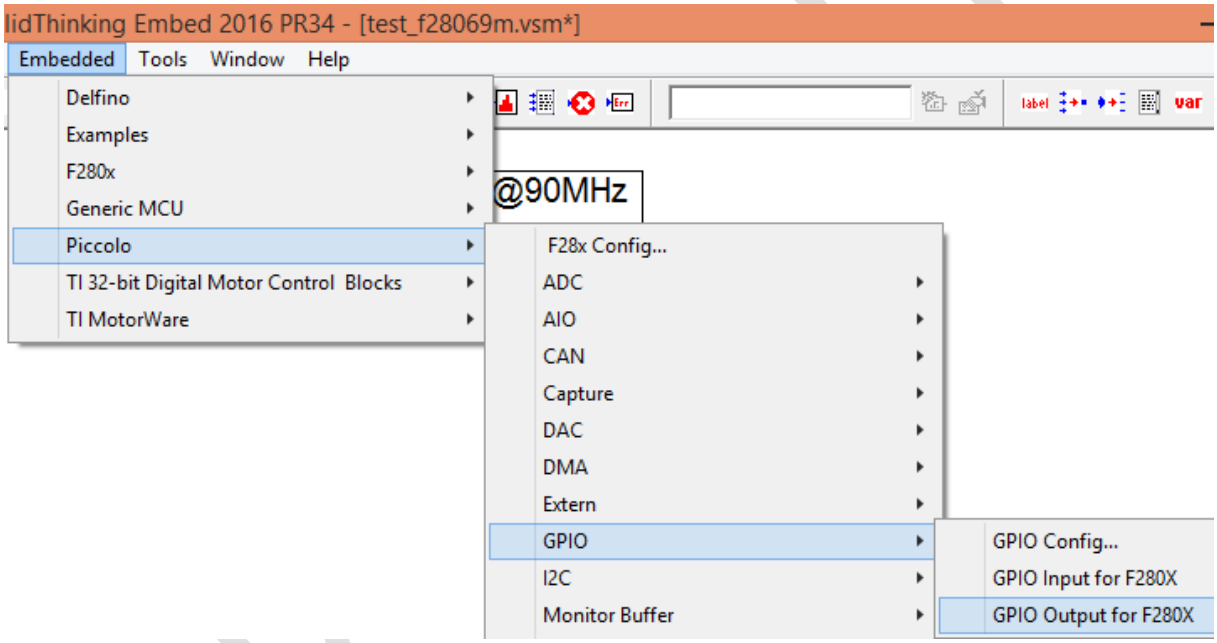


Fig. 51 – Alegerea opțiunii „GPIO Output for f280X”

Pasul 5 – Conceperea unei strategii de comandă:

- Pentru a realiza o aplicație simplă de control al ieșirilor digitale pentru semnalizare intermitentă în mod alternativ, din meniul „Blocks” → „Signal Producer” se alege opțiunea „SquareWave”, iar blocul rezultant se va plasa în spațiul de lucru al modelului. De asemenea, tot din meniu „Blocs” → „Boolean” se va alege opțiunea „not”. Acest bloc realizează funcția de negare sau inversare logică, funcție necesară pentru a controla alternativ ieșirile digitale. Cu ajutorul blocurilor introduse în spațiul de lucru se va realiza următoarea diagramă – model:

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

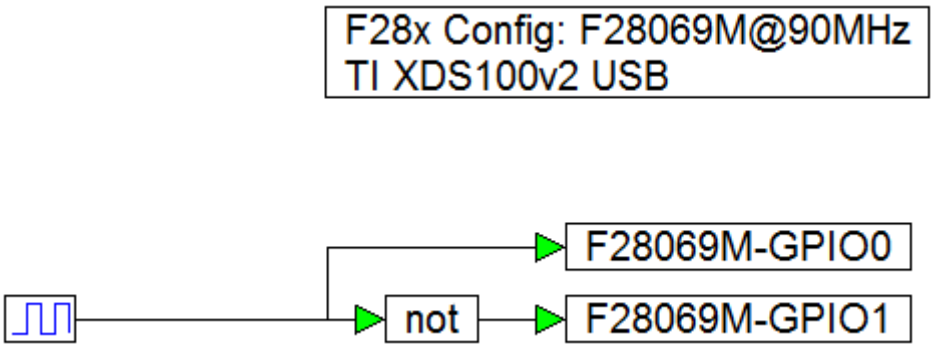


Fig. 52 – Diagramă – model Altair Embed pentru comanda ieșirilor digitale în mod alternativ

Pasul 6 – Generarea automată a programului pe baza strategiei de comandă implementată:

Pentru a genera în mod automat codul sursă al programului care urmează a fi încărcat în memoria platformei de dezvoltare, se vor realiza următorii pași:

- Din meniul „Tools” se va alege opțiunea „Code Gen...” (generare de cod);

În fereastra nou deschisă, se vor introduce următorii parametrii:

- Target: F280X;
- Se va debifa opțiunea „Check for Performance Issues”;

Pentru generare automată de cod, se va apăsa butonul „Code Gen”, pentru vizualizarea codului sursă generat, se va apăsa butonul „View...”, iar pentru generarea fișierului „.out” executabil care va fi încărcat în memoria platformei, se va apăsa butonul „Compile...”. În final, prin intermediul butonului „Download” se va încărca fișierul executabil în memoria platformei.

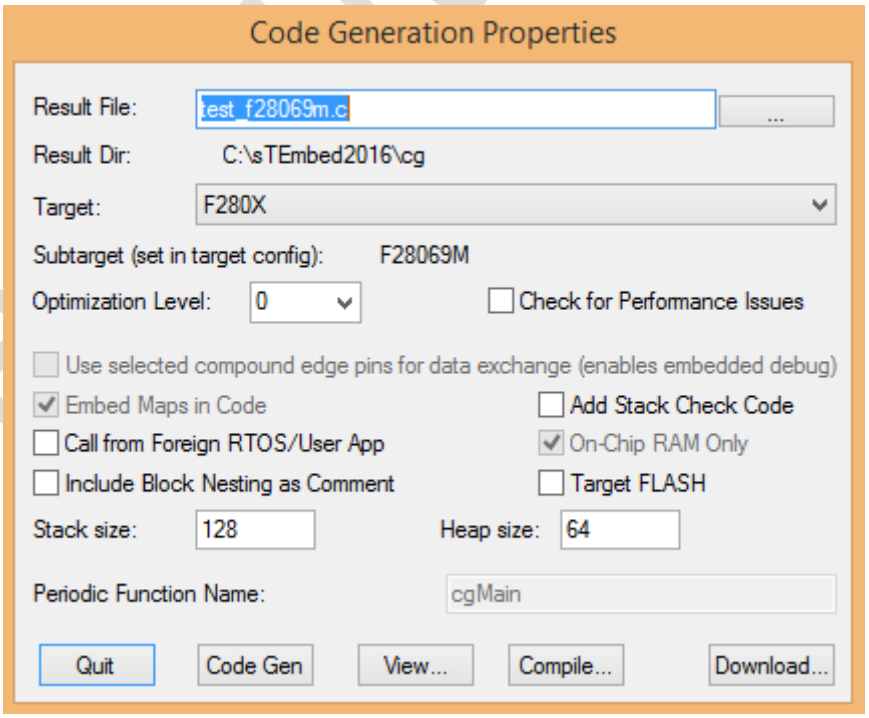


Fig. 53 – Fereastra pentru configurare a generatorului automat de cod

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

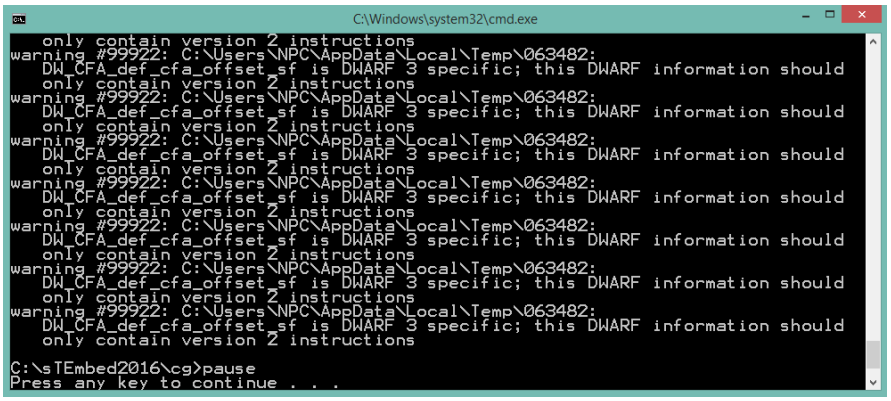


Fig. 54 – Compilarea și generarea fișierului executabil „.out”

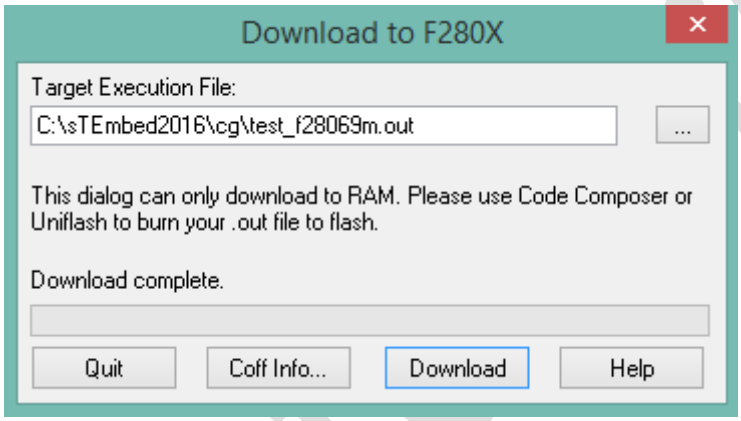


Fig. 55 – Încărcarea fișierului „.out” în memoria RAM a platformei F28069M LaunchPAD

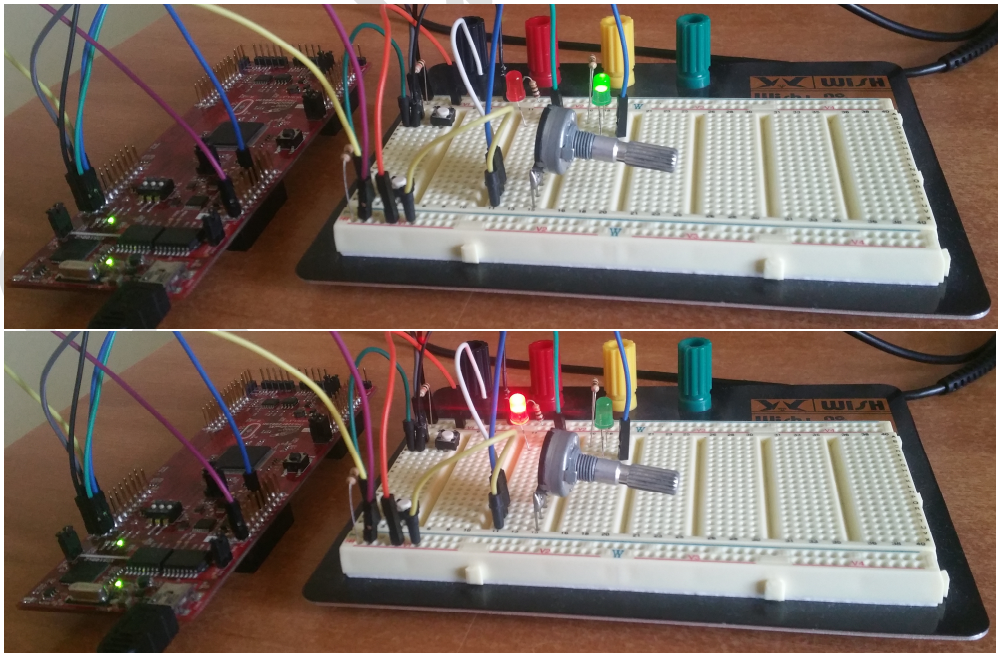


Fig. 56 – Executarea programului în cadrul platformei de dezvoltare

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Codul sursă generat:

```
/*** solidThinking Embed 2016 PR34 Automatic C Code Generator Version 14.0 PR34 ***/
/*  Output for test_f28069m.vsm at Tue Apr 14 19:47:53 2020 */

#include "math.h"
#include "cgen.h"
#include "c2000.h"
int MHZ=90;
#define _SYS_MHZ_ 90

extern CGDOUBLE Zed;

static SIM_STATE tSim={0,0,0
,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0};
SIM_STATE *sim=&tSim;
static INTERRUPT void cgMain()
{
static int _squareCnt4=0;
 int t4;
 PIEACK = 0x1; // Reset PIE IFLG
 t4 = (++_squareCnt4 > 5000?(_squareCnt4>=10000?_squareCnt4=0,1:1):0);
 if ( t4)
  {GPASET = 0x1L;}
 else
  {GPACLEAR = 0x1L;};
 if (!((int) t4))
  {GPASET = 0x2L;}
 else
  {GPACLEAR = 0x2L;};

 endOfSampleCount = TIMER0TIM;

}

void main()
{
 EALLOW;
 CLKCTL &= ~CLK_INTOSC1OFF;       //Internal Osc 1
 CLKCTL &= ~CLK_OSCCLKSRCSEL;
 CLKCTL |= CLK_INTOSC2OFF|CLK_XTALOSCOFF|CLK_XCLKINOFF;
 PLLSTS = 0x10; // reset clk check
 WDCR=0x00ef;         // Disable Watchdog
 asm(" clrc DBGM");
```

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

```
 if (!(PLLSTS&8))       // Skip PLL set if OSC failure
  { PLLSTS = 0x40;      //Disable OSC check
    PLLSTS = 0x180;     //Enable OSC check (&F283xx /2 mode)
    PLLCR = 0x9;        // set PLL to 9xOSC = 90 MHZ;
  }

 PCLKCR3 = 0x100;
 EDIS;
simInit( 0 );
 EALLOW;
 GPADIR = 0x3;
 EDIS;
 startSimDsp();
 installInterruptVec(1,7,&cgMain);
 TIMER0PRD = 0x2328; // 32-bit Timer Period Low
 TIMER0PRDH = 0x0; // 32-bit Timer Period High
 TIMER0TCR |= 0x4020; //Interrupt enable, Timer Reset
 EALLOW;
 PIECTRL = 1; // Enable PIE Interrupts
 EDIS;
 IER |= 0x1; //CPU Interrupt enable
 resetInterrupts();
 enable_interrupts();  // Global Start Interrupts
 EALLOW;
 PCLKCR0 |= 0x4; // Start all PWM timers
 EDIS;
 dspWaitStandAlone();
}
```

MENȚIUNE: Această procedură de implementare, se aplică, în cazul proiectării unor aplicații independente de calculatorul gazdă, sau cu decizie proprie, care funcționează pe baza mărimilor furnizate la intrările platformei (fie digitale, fie analogice).

Pentru a realiza o aplicație de **control și monitorizare în timp real prin intermediul calculatorului gazdă** având ca mijloc de procesar platforma F28069M se va proceda astfel:

A. Realizarea modelului Altair Embed pentru execuție în cadrul platformei (target model):

- Se va creea un model nou cu numele „f28069m_target_test”, în care se vor parcurge în mod similar, pașii inițiali pentru conceperea unui model ca și în cazul anterior;
- În urma parametrizărilor inițiale, se va realiza următorul model:

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

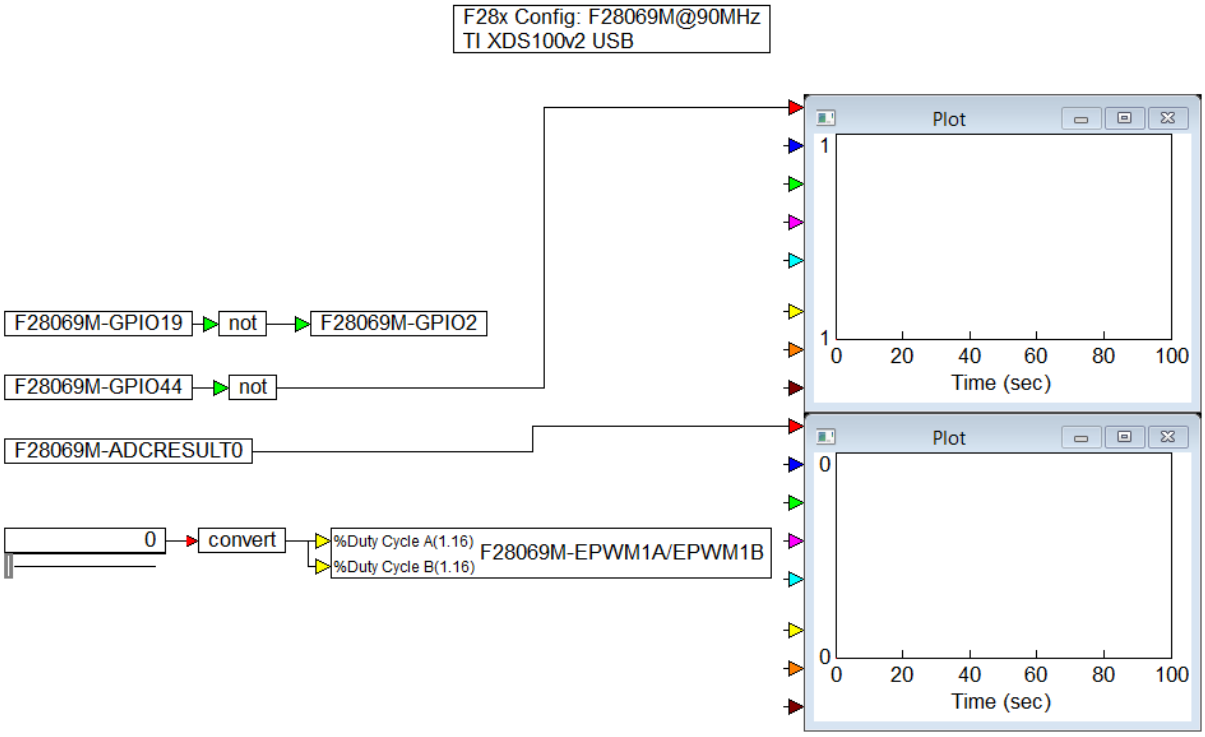


Fig. 57 – Modelul programului ce urmează a fi încărcat în memoria platformei F28069M

- Blocul „Slider” (cursor) se găsește în meniul „Blocks” → „Signal Producer” → „Slider”;
- Blocul „Convert” se găsește în meniul „Blocks” → „Fixed Point” → „Convert”;
- Blocul „not” se găsește în meniul „Blocks” → „Boolean” → „not”;
- Graficul (plot) se găsește în meniul „Blocks” → „Signal Consumer” → „plot”;
- Blocul „ePWM” se găsește în meniul „Embedded” → „Piccolo” → „PWM” → „ePWM”;

În cadrul blocului „ePWM” se vor realiza următoarele parametrizări (dublu click pe bloc):
- Timer Period: 900 (adică 50 kHz – a se vedea documentația introductivă în domeniul DSP);
- EPWMSYNC0: none;
- CMPA Load On: Immediate;
- EPWMSYNC0 pin: Unused;
- CMPN Load On: Immediate
- EPWMA → CMPA: up = 0; down = 1;
- EPWMB → CMPB: up = 0; down = 1;
- Send Start ADC Conversion Pulse A (SOCA): CTR = zero; /1;
- Send Start ADC Conversion Pulse B (SOCB): CTR = PRD; /1;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

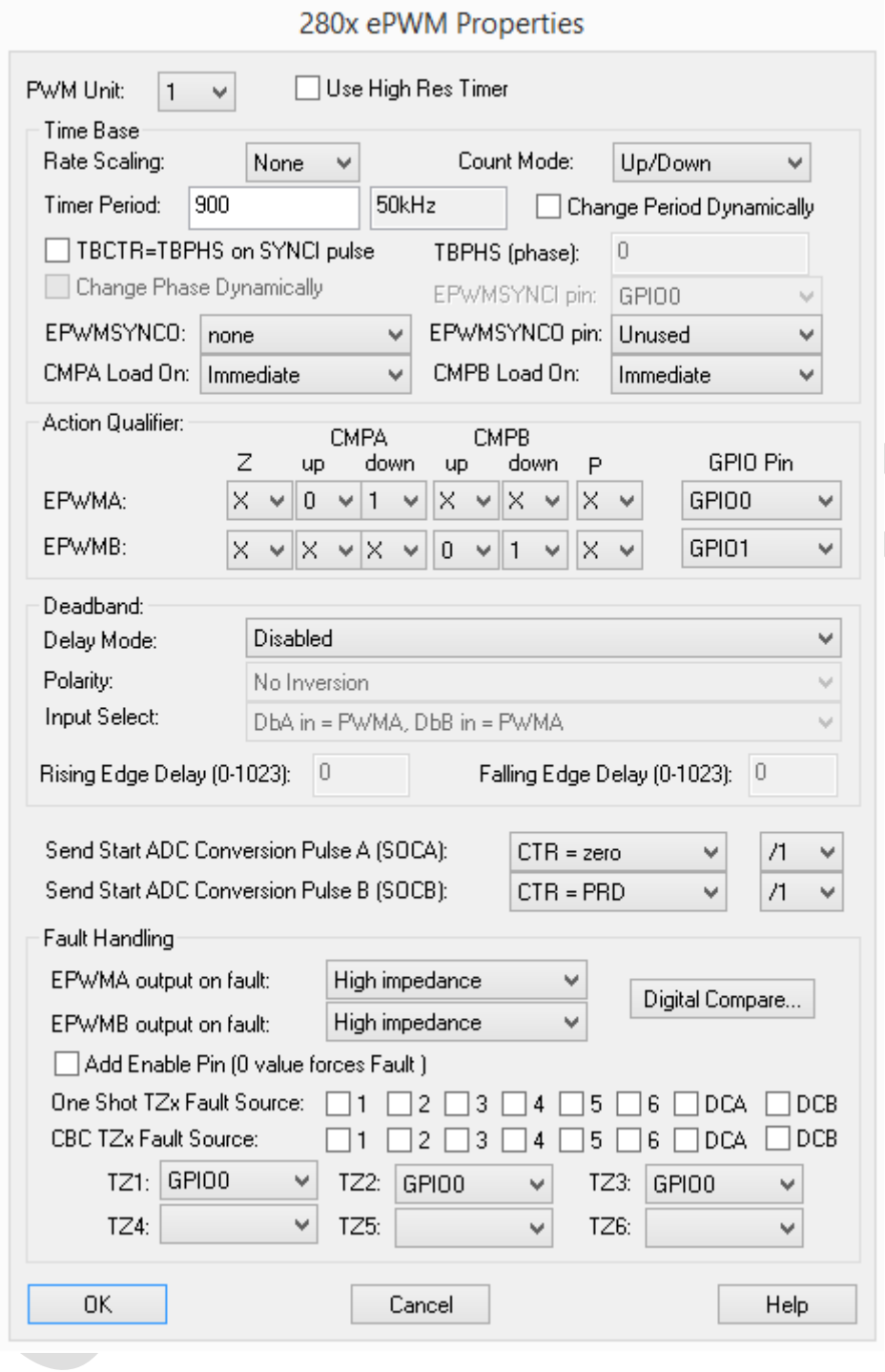


Fig. 58 – Parametrizarea generatorului de semnal modulat în lățime (ePWM)

- Ca și intrări digitale se vor alege pinii „GPIO19” și „GPIO44”;
- Ca și ieșire digitală, se va alege pinul „GPIO2”;
- Ca și intrarea analogică, se va alege registrul „ADCRESULT0”;
- Blocul „ePWM” este utilizează unitatea nr. „1” adică, pinii „GPIO0” și „GPIO1”;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În cadrul blocului „Convert” se vor realiza următoarele parametrizări:
- Tipul de date: Scaled int;
- Radix point (raza de convergență): 1;
- Word Size (dimensiunea de reprezentare): 16 (biți);

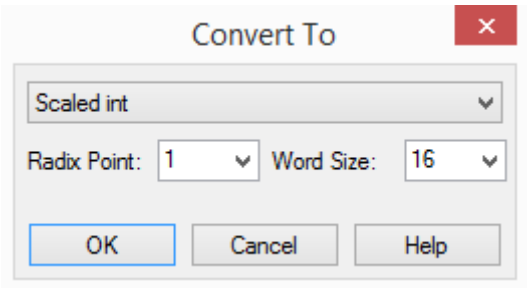


Fig. 59 – Parametrizarea blocului „Convert”

Pentru a stabili corespondența între registrul „ADCRESULT0” și pinul analogic de referință, se vor parametriza setările convertorul analog – digital. Astfel, din meniul „Embedded” → „Piccolo” → „ADC” → „ADC Config...” se vor alege următoarele opțiuni:
- ADCRESULT0 → Src: „A0” → Trigger: „ePWM1-SOCA” → Sample Clks: „7”;

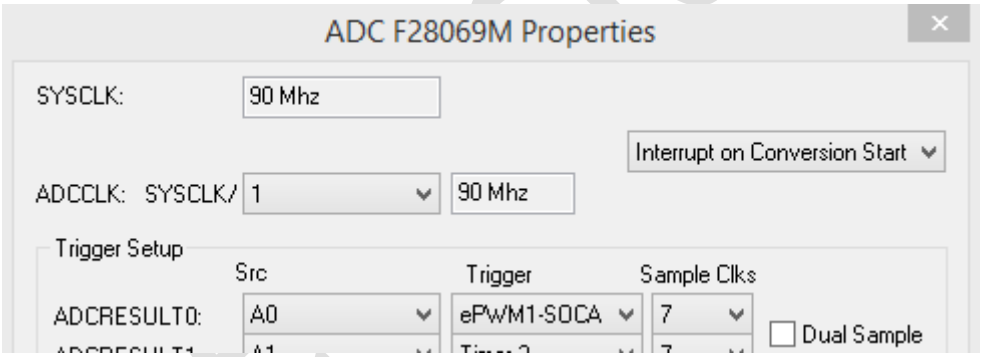


Fig. 60 – Parametrizarea convertorului analog – digital

Pentru a realiza programul care se va încărca în memoria platformei, din modelul creeat, (ținând click stânga apăsat), se vor selecta următoarele blocuri:
- convert;
- not;
- GPIO19, GPIO44, GPIO2;
- ADCRESULT0;
- EPWM;

Cursorul, și graficele vor rămâne neselectate!

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

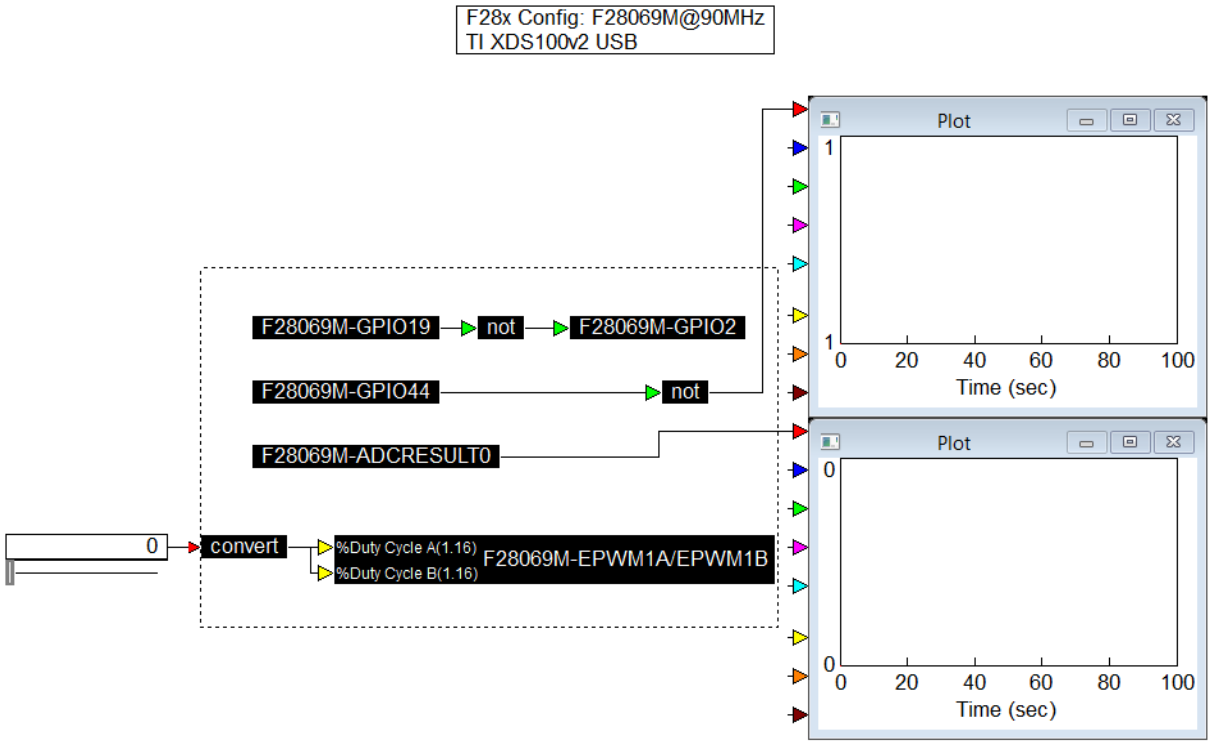


Fig. 61 – Selectarea blocurilor care vor face parte din codul programului

În urma selectării acestor blocrui, (cu ajutorul comenzii „click dreatpa") se va deschide meniul de particularizare al elementelor selectate, și se va alege opțiunea „Create Compound", pentru a creea un sub-sistem.

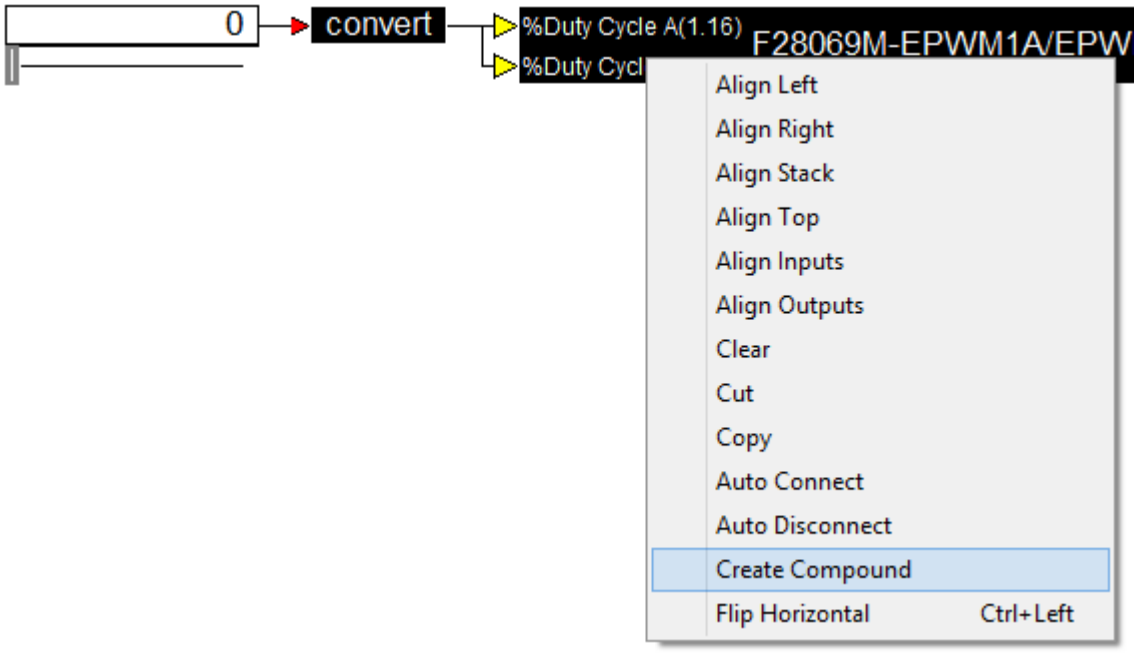


Fig. 62 – Particularizarea elementelor selectate – crearea unui sub-sistem (eng. Compound)

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- În urma acestei operații, va rezulta o casetă de dialog în care există posibilitatea specificării unui nume funcțional pentru sub-sitemul creat. În cazul de față în vom numi „target_code”.

- În vederea generării codului în mod automat, blocul rezultant va fi selectat astfel:

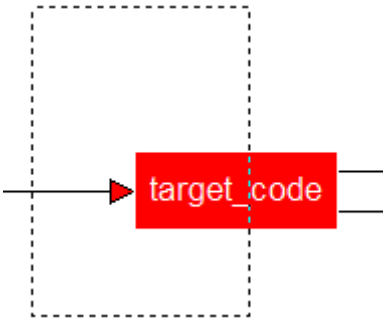


Fig. 63 – Selectarea blocului rezultant în vederea generării codului program

- Având blocul rezultant selectat (culoarea roșie), din meniu „Tools” → „Code Gen...” se va apela fereastra pentru parametrizare a generatorului automat de cod. În cadrul acestei ferestre, se va bifa opțiunea „Use selected compound edge pins for data exchange...”

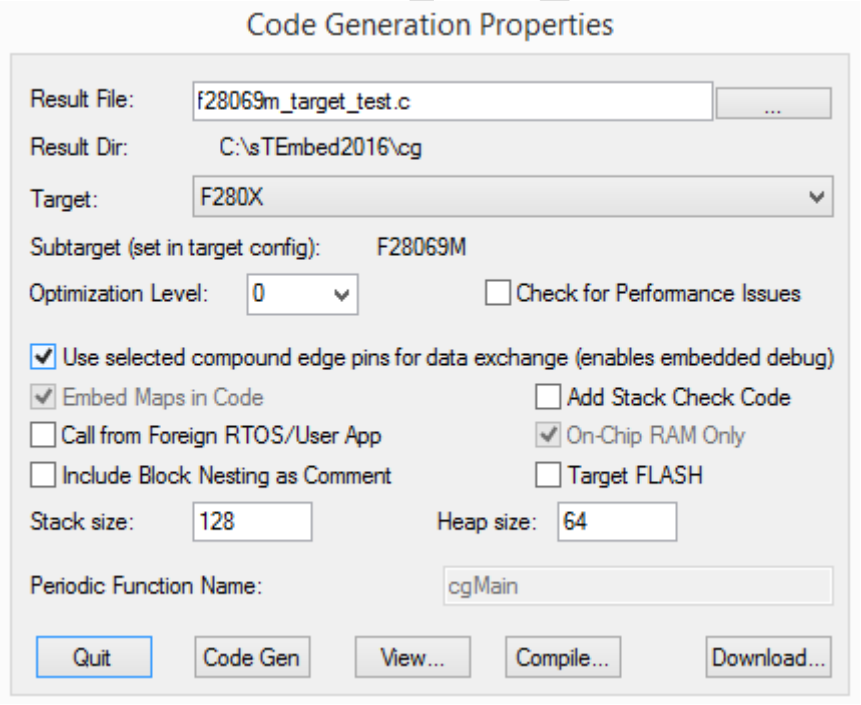


Fig. 64 – Alegerea opțiunii „Use selected compound edge pins for data exchange...”

- În urma acestor acțiuni, se va proceda la generarea automată de cod prin intermediul butonului „Code Gen” apoi, la compilarea codului sursă prin intermediul butonului „Compile”.
- FIȘIERUL EXECUTABIL GENERAT NU SE VA ÎNCĂRCA PRIN INTERMEDIUL BUTONULUI „Download” DE ACEASTĂ DATĂ!

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Codul sursă generat:

```
/*** solidThinking Embed 2016 PR34 Automatic C Code Generator Version 14.0 PR34 ***/
/*  Output for f28069m_target_test.vsm at Tue Apr 14 19:51:58 2020 */

#include "math.h"
#include "cgen.h"
#include "cgendll.h"
#include "c2000.h"
int MHZ=90;
#define _SYS_MHZ_ 90
extern CGDOUBLE Zed;

DLL_SIG_DECL_M(1,2)
static ARG_DESCR outArgInfo13[]={
 { T_INT,0,0,0},
 { T_SCALED_INT,0,4,16},
};
static ARG_DESCR inArgInfo13[]={
 { T_DOUBLE,0,0,0},
};
static SIM_STATE tSim={0,0,0
,outArgInfo13, inArgInfo13,1,2,0,0,0,0,1,1,1,0,0,0,0,0,0};
SIM_STATE *sim=&tSim;
/* target_code */
INTERRUPT void cgMain()
{
 int t8;
 PIEACK = 0x1; // Reset PIE IFLG
  TIMER0TCR = TIMER0TCR;
 t8 = (int)( sim->inSigS[0]->u.Double * 32768);
 if (!((int)((GPADAT & 0x80000L)!=0)))
  {GPASET = 0x4L;}
 else
  {GPACLEAR = 0x4L;};
 { long _duty32 = (long) t8*901;
  CMPA1 = (int)(_duty32>>15);
 }
 CMPB1 = (int)(((long) t8*901)>>15);
 sim->outSigS[0].u.Int = !((int)((GPBDAT & 0x1000L)!=0));
 sim->outSigS[1].u.scaledInt.val = ADCRESULTNWS0;
 endOfSampleCount = TIMER0TIM;
 if (TIMER0TCR&0x8000)
  setTgtStatus(VE_STAT_TGT_OVER_FRAMING);
```

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro

```
}
void main()
{
 EALLOW;
 CLKCTL &= ~CLK_INTOSC1OFF;       //Internal Osc 1
 CLKCTL &= ~CLK_OSCCLKSRCSEL;
 CLKCTL |= CLK_INTOSC2OFF|CLK_XTALOSCOFF|CLK_XCLKINOFF;
 PLLSTS = 0x10; // reset clk check
 WDCR=0x00ef;         // Disable Watchdog
 asm(" clrc DBGM");
 if (!(PLLSTS&8))         // Skip PLL set if OSC failure
  { PLLSTS = 0x40;      //Disable OSC check
    PLLSTS = 0x180;     //Enable OSC check (&F283xx /2 mode)
    PLLCR = 0x9;        // set PLL to 9xOSC = 90 MHZ;
  }
 HISPCP = 0x0;    // HCLK  = 90 MHZ
 PCLKCR0 = 0x8;
 PCLKCR1 = 0x1;
 PCLKCR3 = 0x100;
 EDIS;
 TBPRD1 = 0x384;
 AQCTLA1 = 0x90;
 AQCTLB1 = 0x900;
 CMPCTL1 = 0x50;
 ETSEL1 = 0xa900;
 ETPS1 = 0x1100;
 EALLOW;
 TZCTL1 = 0x0;
 TZSEL1 = 0x0;
 DCACTL1 = 0x0;
 DCBCTL1 = 0x0;
 EDIS;
simInit( &tSim );
 EALLOW;
 Device_cal(); // Call on chip calibration
 EALLOW;
 ADCCTL1 = 0x40E0;  // Power up ADC
 ADCSOC0CTL = 0x2806; // ADCINA0, SOC = ePWM1:ADCSOCA
 EDIS;
 EALLOW;
 GPAMUX1 = 0x5;
 GPADIR = 0x4;
 EDIS;
 startSimDsp();
```

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

epe.utcluj.ro

```
 installInterruptVec(1,7,&cgMain);
 TIMER0PRD = 0x2328; // 32-bit Timer Period Low
 TIMER0PRDH = 0x0; // 32-bit Timer Period High
 TIMER0TCR |= 0x4020; //Interrupt enable, Timer Reset
 EALLOW;
 PIECTRL = 1; // Enable PIE Interrupts
 EDIS;
 IER |= 0x1; //CPU Interrupt enable
 resetInterrupts();
 disable_interrupts();  // Disable interrupts until PC handshake complete
 TBCTL1 = 0x32; // Start timer
 EALLOW;
 PCLKCR0 |= 0x4; // Start all PWM timers
 EDIS;
 dspWait();
}
```

B. Realizarea modelului de control Altair Embed pentru calculatorul gazdă (host model):

- Pentru a realiza modelul de control Altair Embed care să ruleze pe calculatorul gazdă, se va salva actualul model cu o nouă denumire, anume „f28069m_host_test”;

- Din cadrul meniului „Embedded” → „Piccolo” → „Target Interface”, se va selecta opțiunea „Target Interface”, iar blocul rezultant acestei operații, va fi plasat în model. Totodată, în cadrul parametrilor acestui bloc, se va debifa opțiunea „Show CPU Utilization”;

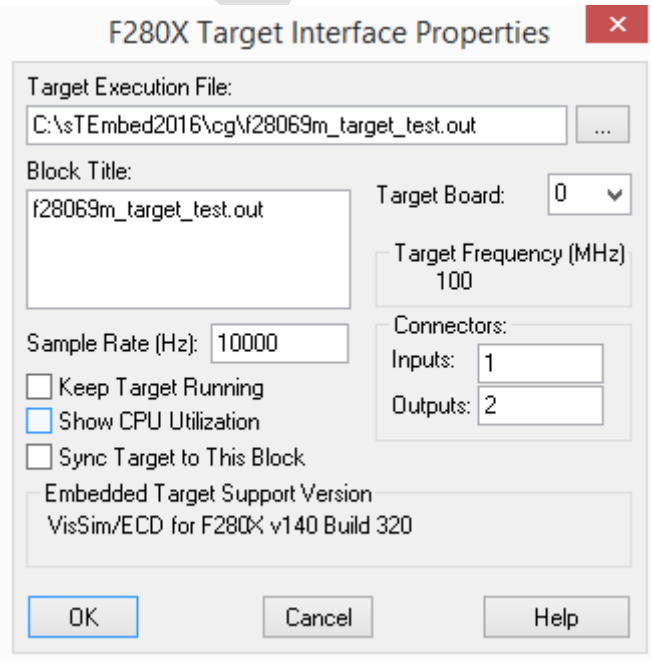


Fig. 65 – Parametrizarea blocului „Target Interface”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

emd.utcluj.ro

- Se vor conecta terminalele blocului „Target Interface” în mod similar conexiunilor blocului „target_model”: la intrarea se va lega cursorul, la ieșire, cele două grafice:

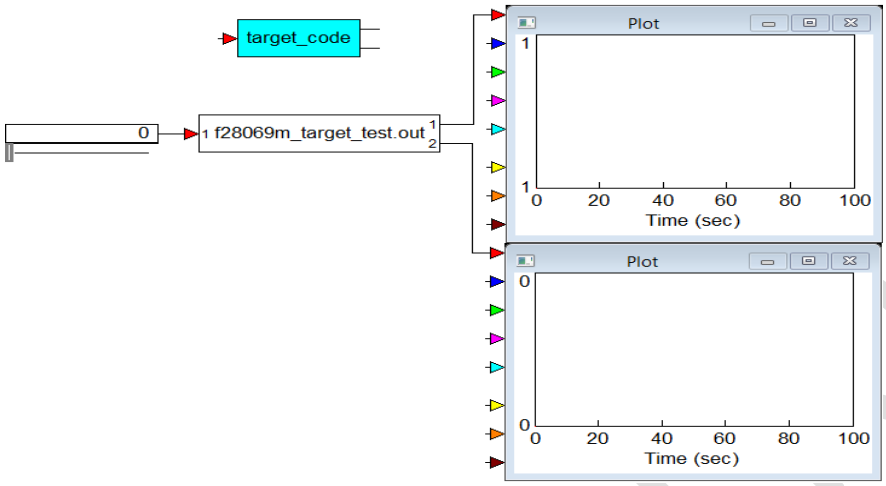


Fig. 66 – Modelul de control Altair Embed pentru calculatorul gazdă

- Pentru a lansa în execuție aplicația în timp real, din meniul „System” se va alege opțiunea „Go”. În urma acestei operații, se va încărca fișierul executabil în memoria RAM a platformei:

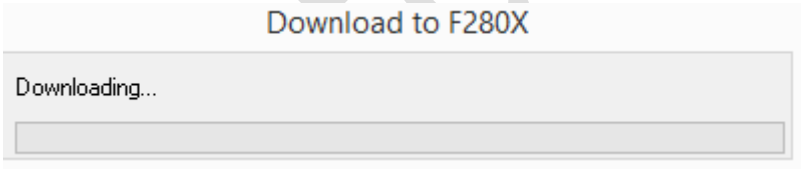


Fig. 67 – Încărcarea fișierului executabil în memoria platformei F28069M

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

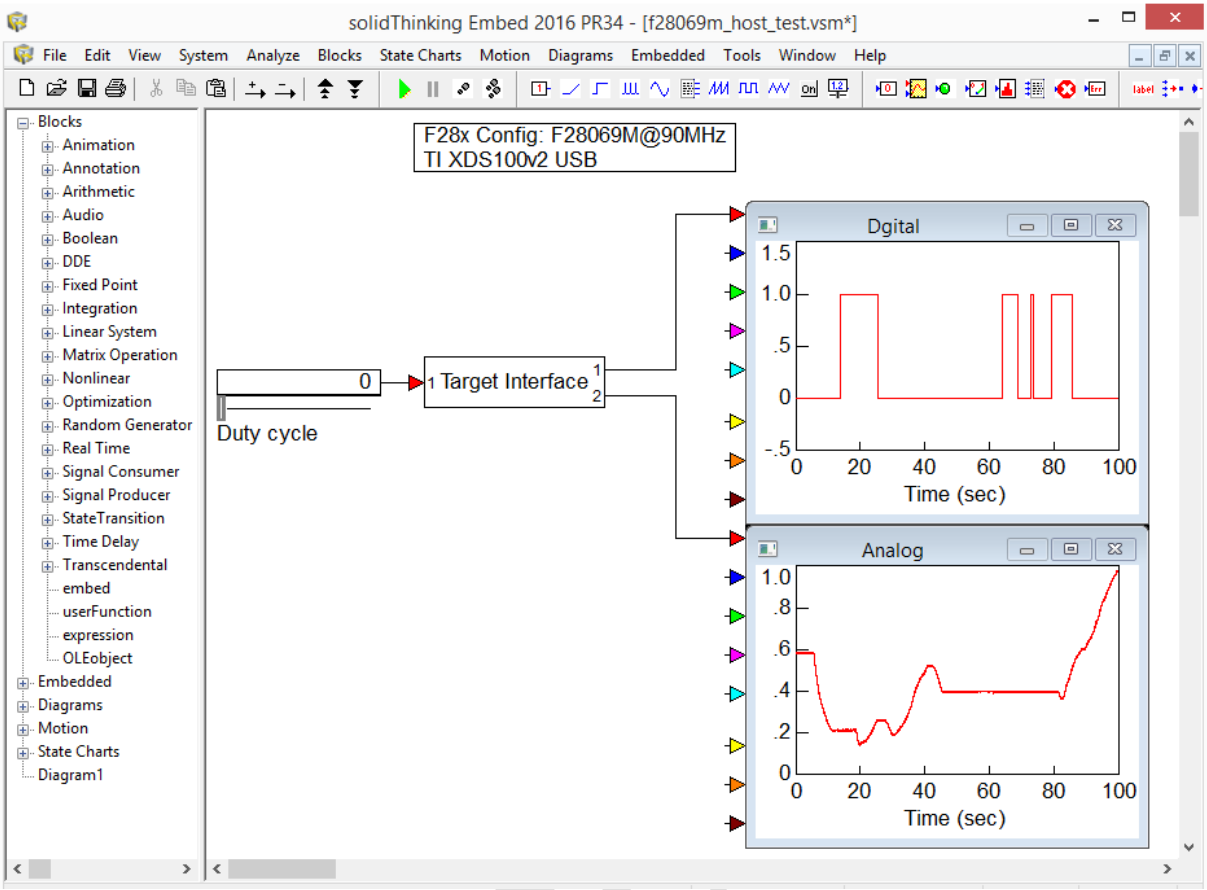


Fig. 68 – Rularea modelului în timp real atât pe platforma F28069M cât și pe calculator

Prin intermediul modelului conceput, se pot realiza următoarele funcții:

- citirea și afișarea în calculator a unei mărimi analogice de la potențiometru;

- citirea și afișarea în calculator a unei mărimi digitale de la buton;

- direcționarea stării intrării digitale înspre o ieșire digitală (LED verde controlat de buton);

- variația factorului de umplere pentru unitatea PWM aferentă unei ieșirie digitale (LED roșu);

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

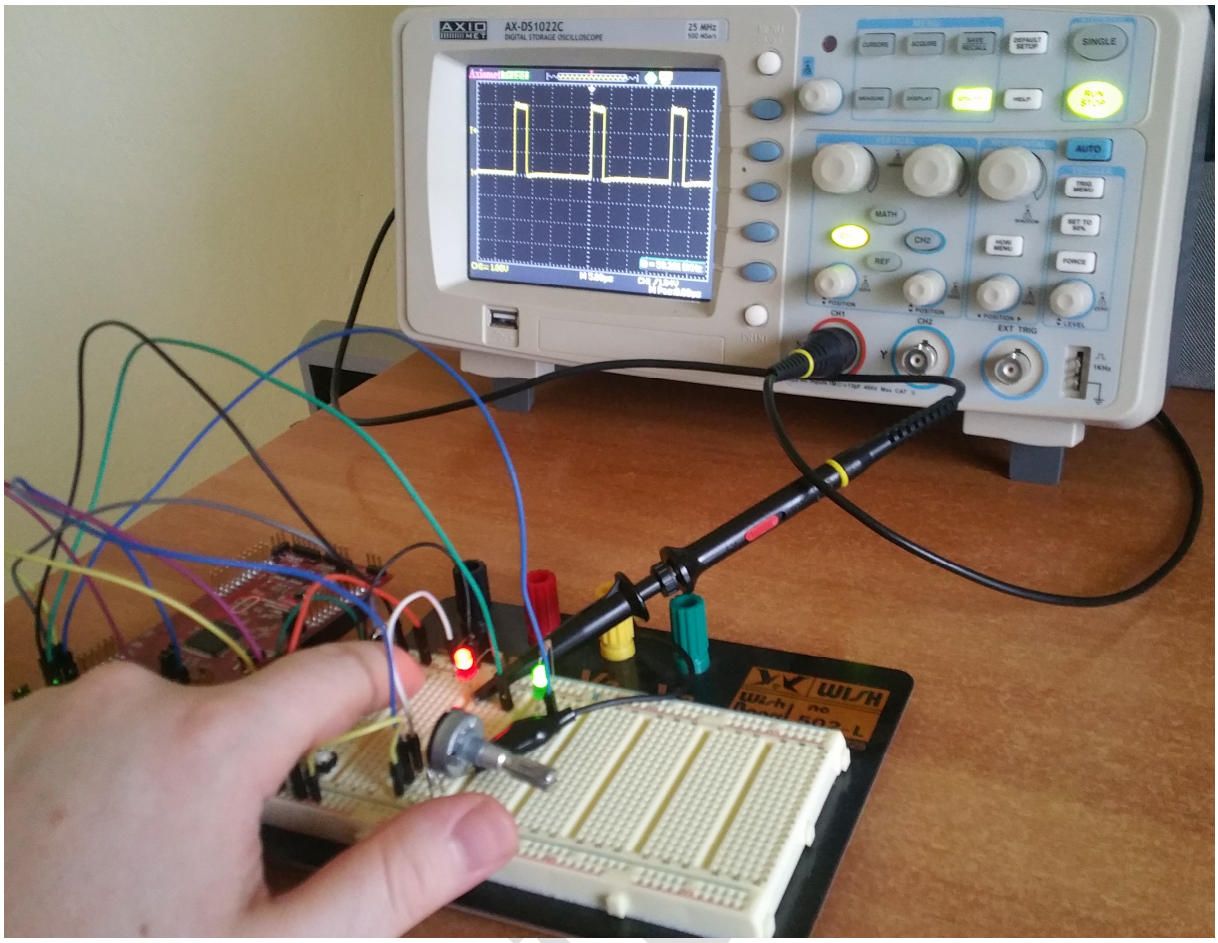


Fig. 69 – Funcționare integrală a montajului pentru aplicația executată în timp real

III. CONCLUZII:

- În vederea dezvoltării unei aplicații complexe, sistemele de calcul bazate pe procesoare digitale de semnal (eng. DSP), necesită nu doar un simplu mediu de programare, ci și un mediu de testare și simulare în timp real (ex. Altair / SolidThinking Embed, Matlab – Simulink etc...);

- Generarea automată a codului sursă pentru o aplicație, în cazul DSP, este mult mai facilă!

- Platformele de dezvoltare cu DSP Texas Instruments, care dispun de un adaptor jTAG – USB încorprorat (precum F28069M LaunchPAD), reprezintă o soluție compactă, având cost redus și oferă o largă varietate de posibilități de programare și testare a codului sursă generat.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

IV. BIBLIOGRAFIE:

1. Teodor Crișan Pană – „Sisteme de calcul cu microprocesoare, FPGA și DSP” – Editura UTPRESS, Cluj – Napoca, 2016 – ISBN 978-606-737-206-9;

2. Ioana – Cornelia GROS, Lucian – Nicolae PINTILIE, Teodor Crișan PANĂ – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAȚII” – Editura UTPress Cluj – Napoca, 2020 ISBN 978-606-737-431-5: (https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/431-5.pdf);

3. Texa Instruments F28069M LaunchPAD datasheet: http://www.ti.com/lit/ug/sprui11b/sprui11b.pdf

4. Texas Instruments - Catalog TMS320F2806x: (http://www.ti.com/lit/ds/sprs698h/sprs698h.pdf);

5. Texas Instruments – Manual tehnic de referință: (http://www.ti.com/lit/ug/spruh18h/spruh18h.pdf);

6. Texas Instruments Company – „Code Composer Studio v5 Users Guide”: https://processors.wiki.ti.com/index.php/Code_Composer_Studio_v5_Users_Guide

7. Material video – „TI C2000 LaunchPad F28069: Tutorial 0 Flash”: https://www.youtube.com/watch?v=62m0lUxBwp8

8. MathWorks Company – „Embedded IDE Link™ CC 3 User’s Guide”: http://www.eas.uccs.edu/~mwickert/ece5655/lecture_notes/ccslink.pdf

9. Altair Embed / SolidThinking Embed / VisSim User Guide: http://www.vissim.us/downloads/doc/VisSim_UGv90.pdf

10. Material video – „Altair Embed Training - PWM and ADC Use and Synchronization”: https://www.youtube.com/watch?v=jQ1I7MBooBg

11. Texas Instruments – „Real-Time Data Exchange” White paper: SPRY012: https://e2echina.ti.com/cfs-file/__key/telligent-evolution-components-attachments/13-106-00-00-00-00-41-26/Real_2D00_Time-Data-Exchange.pdf

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

ee.utcluj.ro