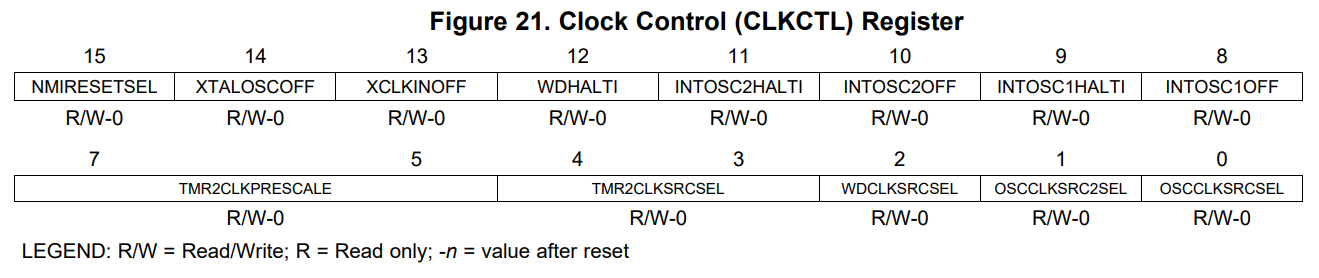
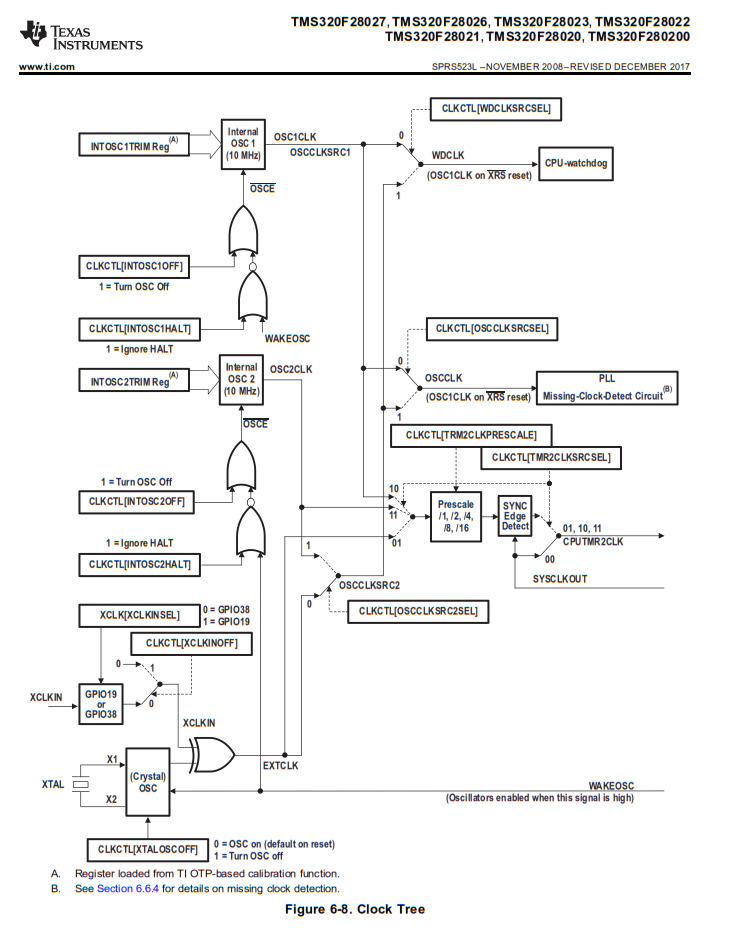
TÀI LIỆU NÀY ĐƯỢC VIẾT RA NHẰM MỤC ĐÍCH RÚT NGẮN THỜI GIAN CHO NGƯỜI MỚI BẮT ĐẦU LÀM VIỆC VỚI DÒNG TMS320, CÁC HƯỚNG DẪN CHI TIẾT CỤ THỂ ĐỂ CÓ THỂ TIẾN HÀNH CẤU HÌNH VÀ CHẠY ĐƯỢC CÁC CHỨC NĂNG.

1. **Cấu hình Clock cho CPU**

Thông qua thanh ghi CLKCTL(Clock Control Register) ta sẽ cấu hình Clock cho CPU.



Sơ đồ nguyên lý.



Các bit trong thanh ghi CLKCTL

Bit 15- NMIRESETSEL-

Bit 14- XTALOSCOFF: 0-Sử dụng bộ dao động tinh thể XTAL(Thạch anh ngoài), 1-Không sử dụng bộ dao động tinh thể XTAL.

Bit 13- XCLKINOFF: 0-Đầu vào dao động XCLKIN(PIN23/GPIO38) bật, 1-Đầu vào dao động tắt

Bit 12- WDHALTI-

Bit 11- INTOSC2HALTI-

Bit 10- INTOSC2OFF: 0-Bật bộ dao động nội INTOSC2, 1-Tắt bộ dao động nội INTOSC2

Bit 9- INTOSC1HALT: Hệ số nhân cho bộ Timer2, 000-1,001-2,010-4,011-8,100-16

Bit 8- INTOSC1OFF: 0-Bật bộ dao động nội INTOSC1, 1-Tắt bộ dao động nội INTOSC1

Bit 7..5- TMR2CLKPRESCALE-

Bit 4..3- TMR2CLKSRCSEL: Chọn nguồn Clock cho timer 2, 0-Nguồn SYSCLKOUT(clock cấp cho hệ thống, sau khi đã được cấu hình), 01-Nguồn dao động ngoài, 10-Nguồn dao động nội 1 INTOSC1, 11-Nguồn dao động nội 2 INTOSC2

Bit 2- WDCLKSRCSEL: Chọn bộ dao động cấp cho watchdog, 0-Lựa chọn bộ dao động nội 1 INTOSC1, 1-Lựa chọn bộ dao động nội 2 INTOSC2 hoặc bộ dao động ngoại XTAL.

Bit 1- OSCCLKSRC2SEL: 0-Lựa chọn bộ dao động ngoài (XTAL), 1-Lựa chọn bộ dao động nội

Bit 0- OSCCLKSRCSEL: 0-Lựa chọn bộ dao động nội INTOSC1, 1-Lựa chọn bộ dao động nội INTOSC2 hoặc bộ dao động ngoại XTAL.

Cấu hình Clock cho trường hợp sử dụng thạch anh ngoài XTAL

Bit 15- NMIRESETSEL: default

Bit 14- XTALOSCOFF: sử dụng bộ dao động ngoại nên set = 0

Bit 13- XCLKINOFF: không sử dụng đầu vào này nên set = 1

Bit 12- WDHALTI: default

Bit 11- INTOSC2HALTI: default

Bit 10- INTOSC2OFF: không sử dụng bộ dao động nội 2 nên set =1

Bit 9- INTOSC1HALT: hệ số nhân cho timer 2 để mặc định

Bit 8- INTOSC1OFF: có sử dụng bộ dao động nội 1 nên set = 0

Bit 7..5- TMR2CLKPRESCALE: default

Bit 4..3- TMR2CLKSRCSEL: sử dụng nguồn SYSCLKOUT nên set =0

Bit 2- WDCLKSRCSEL: sử dụng nguồn dao động nội 1 INTOSC1 nên set =0

Bit 1- OSCCLKSRC2SEL: sử dụng nguồn dao động ngoài nên set =0

Bit 0- OSCCLKSRCSEL: sử dụng nguồn dao động ngoài nên set =1

Cấu hình bằng phần mềm, sử dụng bộ dao động ngoài XTAL

EALLOW;

SysCtrlRegs.CLKCTL.bit.XTALOSCOFF = 0; // Turn on XTALOSC

SysCtrlRegs.CLKCTL.bit.XCLKINOFF = 1; // Turn off XCLKIN

SysCtrlRegs.CLKCTL.bit.OSCCLKSRC2SEL = 0; // Switch to external clock

SysCtrlRegs.CLKCTL.bit.OSCCLKSRCSEL = 1; // Switch from INTOSC1 to INTOSC2/ext clk

SysCtrlRegs.CLKCTL.bit.WDCLKSRCSEL = 0; // Clock Watchdog off of INTOSC1

SysCtrlRegs.CLKCTL.bit.INTOSC2OFF = 1; // Turn off INTOSC2

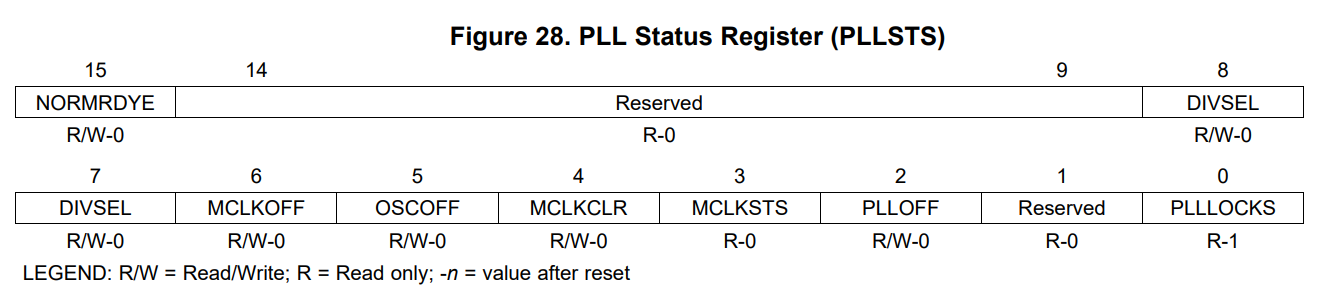
SysCtrlRegs.CLKCTL.bit.INTOSC1OFF = 0; // Leave INTOSC1 on

EDIS;

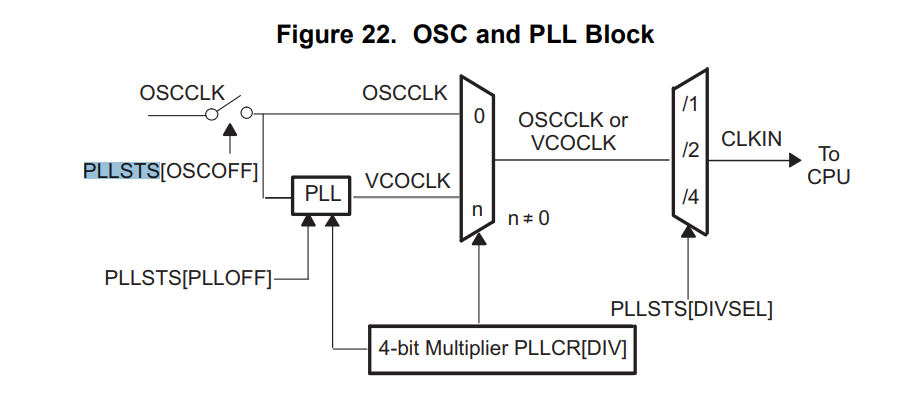
Hoặc có thể sử dụng thư viện có sẵn trong DSP2802x\_SysCtrl.c/XtalOscSel();

1. **Cấu hình bộ PLL**

Thông qua thanh ghi PLLSTS(PLL Status Register) ta sẽ cấu hình bộ nhân tần số cho CPU.



Sơ đồ nguyên lý



Các bit trong thanh ghi PLLSTS

Bit 15- NORMRDYE:

Bit 14..9- Reserved:

Bit 8..7- DIVSEL: Chia hệ số cho CLKIN, 00/01 chia 4, 10 chia 2, 11 chia 1, ban đầu mặc định là 00

Bit 6- MCLKOFF: Phát hiện đồng hồ bị thiếu, 0-Bật logic phát hiện đồng hồ bị thiếu, 1-Tắt logic phát hiện đồng hồ bị thiếu, khi khởi tạo lần đầu thì phải set=1.

Bit 5- OSCOFF

Bit 4- MCLKCLR

Bit 3- MCLKSTS: Kiểm tra trạng thái đồng hồ bit, 0-Nếu bình thường, 1-OSCCLK bị phát hiện là bị thiếu, ghi vào MCLKCLR giá trị 1

Bit 2- PLLOFF

Bit 1- Reserved

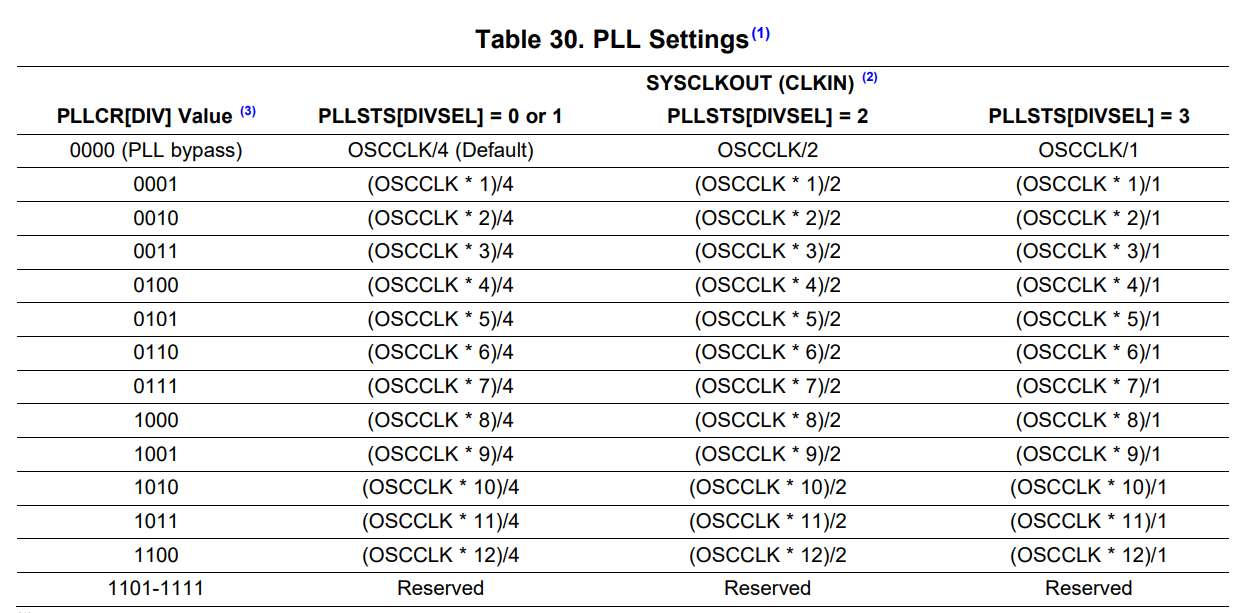
Bit 0- PLLLOCKS: Bit trạng thái khóa PLL, 0-Cho biết thanh ghi PLLCR đã được ghi vào PLL và hiện đang khóa, 1-Chỉ ra rằng PLL đã được khóa và giờ ổn định.

Cấu hình phần mềm.

Sử dụng thạch anh ngoài, tần số 20MHZ

Tần số hệ thống mong muốn 60MHZ

Công thức tính tần số dao động sau khi gọi hàm InitPLL.



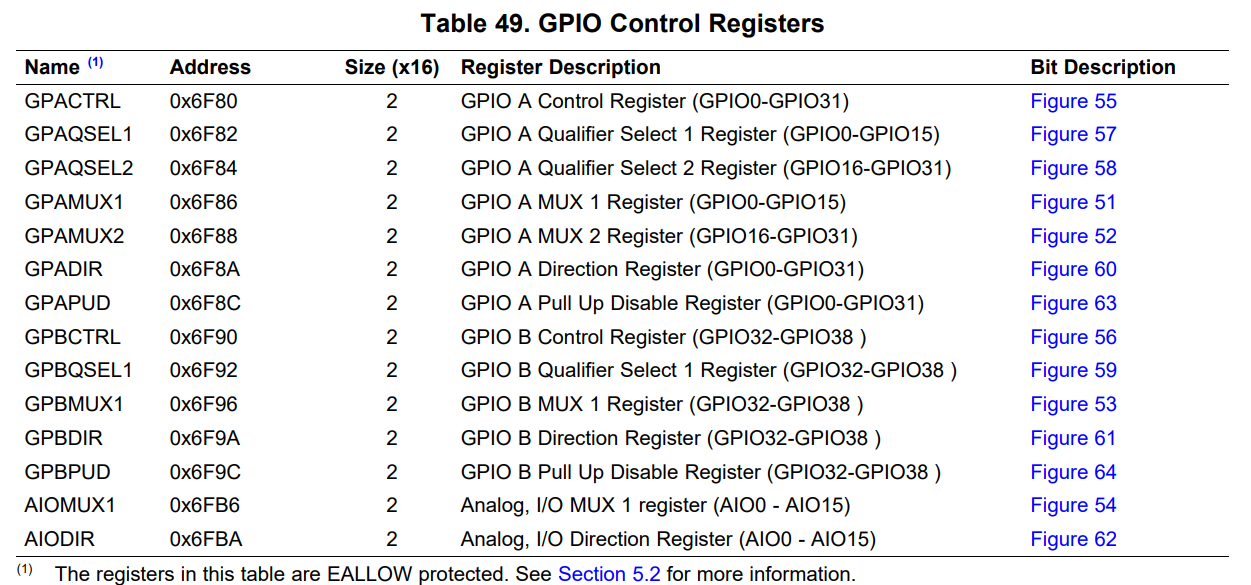
Để thỏa mãn điều kiện đầu ra mong muốn ta sẽ đặt val=0b0110,divsel=2(0b10)

SYSCLKOUT=20MHZ\*6/2=60MHZ

Trong đoạn code DSP2802x\_SysCtrl.c sẽ sửa lại thành .

InitPll(0b0110,0b10);

1. **Cấu hình chân GPIO**



Cấu hình GPIO làm chân IO/PWM thông qua thanh ghi GPAMUX1, GPAMUX2, GPBMUX1

Cấu hình GPIO làm chân Input/Output thông qua thanh ghi GPADIR,GPBDIR

Khi cấu hình GPIO làm chân Output, define GPACLEAR để đưa mức logic low, define GPASET để đưa mức logic high.

Khi cấu hình GPIO làm chân Input, kiểm tra GPADAT/GPBDAT để kiểm tra mức logic của đầu vào.

Ví dụ set chân GPIO0 làm đầu ra, mức logic mặc định là 0.

// GPIO

GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0; // 0=GPIO, 1=EPWM1A, 2=Resv, 3=Resv

GpioCtrlRegs.GPADIR.bit.GPIO0 = 1; // 1=OUTput, 0=INput

GpioDataRegs.GPACLEAR.bit.GPIO0 = 1; // uncomment if --> Set Low initially

// GpioDataRegs.GPASET.bit.GPIO0 = 1; // uncomment if --> Set High initially

Ví dụ set chân GPIO0 làm đầu vào, kiểm tra mức logic của chân.

// GPIO

GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0; // 0=GPIO, 1=EPWM1A, 2=Resv, 3=Resv

GpioCtrlRegs.GPADIR.bit.GPIO0 = 0; // 1=OUTput, 0=INput

// GpioDataRegs.GPACLEAR.bit.GPIO0 = 1; // uncomment if --> Set Low initially

// GpioDataRegs.GPASET.bit.GPIO0 = 1; // uncomment if --> Set High initially

// kiểm tra mức logic của chân GPIO0

if(GpioDataRegs.GPBDAT.bit.GPIO0==0){

}

else {

}

1. **Cấu hình PWM**

**Tài liệu tham khảo: TMS320x2802x, 2803x Piccolo Enhanced Pulse**

**Width Modulator (ePWM) Module**

Chú ý khi cấu hình PWM thì phải set Clock source cho PWM

// SysCtrlRegs.PCLKCR1.bit.EPWM1ENCLK = 1; // EPWM1

// SysCtrlRegs.PCLKCR1.bit.EPWM2ENCLK = 1; // EPWM2

SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1; // EPWM3

// SysCtrlRegs.PCLKCR1.bit.EPWM4ENCLK = 1; // EPWM4

Trong đoạn code trên ta sử dụng ePWM3 thì phải set EPWM3ENCLK=1

Cách thanh ghi được sử dụng trong cấu hình PWM

TBCTL

TBPRD

TBPHS

TBCTR

CMPCTL

AQCTLA

AQCTLB

CMPA

CMPB

DBCTL

DBRED

DBFED

TZSEL

TZCTL

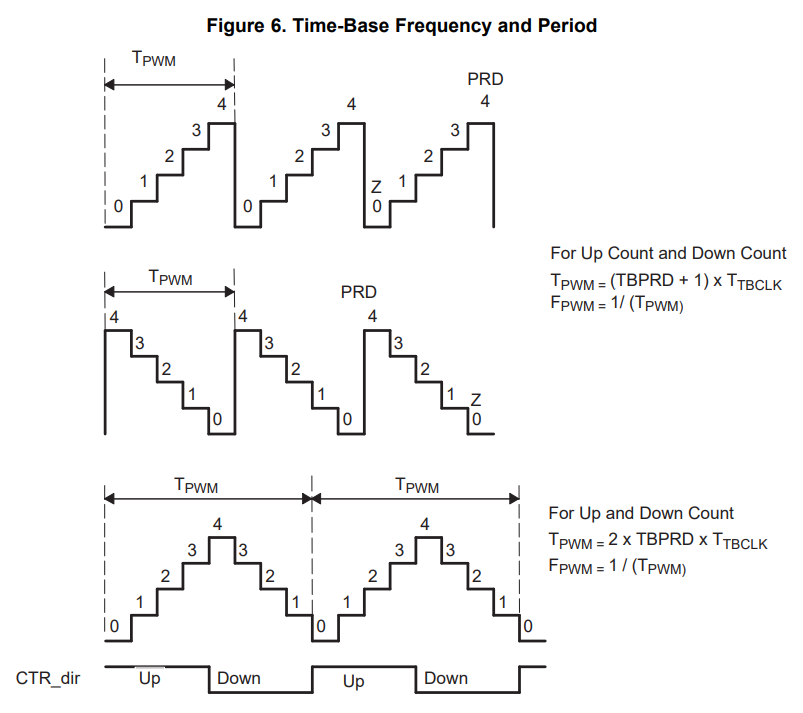
Ta chỉ cần quan tâm đến chức năng của những bit đã được cấu hình trong thanh ghi ở trong bộ thư viện đã có.

(\*ePWM[j]).TBCTL.bit.PRDLD = TB\_IMMEDIATE; // IMMEDIATE mode

Immediate mode means whatever value you wrote takes effect at that very moment.  Shadow mode means it will not take effect until some defined time - for example, when a timer is equal to zero.

(\*ePWM[j]).TBPRD = PRDValue; // PWM frequency = 1/(2\*TBPRD)

Tính toán tần số ePWM



TBCLK Time base clock

TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)

HSPCLKDIV=1;(set bit tương ứng =0/tra datasheet)

CLKDIV=1; (set bit tương ứng =0/tra datasheet)

Đặt tần số PWM=80Khz, tần số cơ sở TBCLK = 60Mhz

Chế độ mode 3

Giá trị TBPRD=Tpwm/(2\*TBCLK)=(1/80000)/(2\*1/60000000)=375

(\*ePWM[j]).TBPRD = 375;

**Cấu hình ngắt PWM**

Muốn tạo ngắt PWM ta phải khai báo như sau.

Khai báo hàm khi có ngắt PWM xảy ra thì sẽ nhảy vào đó để xử lý.

**interrupt** **void** **epwm3\_isr**(**void**);

khi xảy ra ngắt thì sẽ nhảy vào trong hàm này, cần phải gán địa chỉ của hàm này vào bảng vecto để khi có ngắt thì sẽ nhảy vào đó.

PieVectTable.EPWM3\_INT = &epwm3\_isr;

Khi xảy ra ngắt thì sẽ nhảy đến địa chỉ EPWM3\_INT đã được gán và thực hiện chương trình ở đó

Trong trường hợp này ta khai báo trong bảng vecto ngắt là PieVectTable.EPWM3\_INT tương ứng với ngắt timer, vậy ta cần phải cấu hình ngắt timer để có thể thực hiện được ngắt và nhảy vào trong hàm xử lý.

// Enable Timer interrupt

EPwm3Regs.ETSEL.bit.INTSEL = ET\_CTR\_ZERO; // Enable INT on Zero event

EPwm3Regs.ETSEL.bit.INTEN = 1; // Enable INT, cho phép ngắt

EPwm3Regs.ETPS.bit.INTPRD = ET\_1ST;// Generate INT on 1rd event, nhảy vào hàm xử lý ngắt PWM khi có ngắt PWM xảy ra

// ET\_2ND,ET\_3ND, sau 2, 3 chu kỳ ngắt PWM thì sẽ nhảy vào hàm vecto xử lý ngắt PWM

Thanh ghi ETSEL Event Trigger Selection Register.

Khai báo nhóm vecto ngắt.

Bảng vecto ngắt sẽ kiểm tra ngắt nào được kích hoạt thì sẽ cho tiến hành xử lý ngắt tương ứng.

Để biết ngắt PWM3 được khai báo như nào ta tìm tới file .device\_support/f2802x/v210/f2802x\_common/source/DSP2802x\_SWPrioritizedDefaultIsr.c

Ta tìm hàm EPWM3\_INT\_ISR tương ứng với khai báo cho ngắt timer PWM 3

// Connected to PIEIER3\_3 (use MINT3 and MG33 masks):

#if (G33PL != 0)

interrupt void EPWM3\_INT\_ISR(void) // EPwm3 Interrupt

{

// Set interrupt priority:

volatile Uint16 TempPIEIER = PieCtrlRegs.PIEIER3.all;

IER |= M\_INT3;

IER &= MINT3; // Set "global" priority

PieCtrlRegs.PIEIER3.all &= MG33; // Set "group" priority

PieCtrlRegs.PIEACK.all = 0xFFFF; // Enable PIE interrupts

EINT;

// Insert ISR Code here.......

// Restore registers saved:

DINT;

PieCtrlRegs.PIEIER3.all = TempPIEIER;

// Next two lines for debug only to halt the processor here

// Remove after inserting ISR Code

asm (" ESTOP0");

for(;;);

}

#endif

Ta sẽ thấy các thông số sau IER |= M\_INT3; và PieCtrlRegs.PIEIER3.all &= MG33;// (MG33: số 3 đầu PIEIER3, số 3 tiếp INTx3, ứng với PieCtrlRegs.PIEIER3.bit.INTx3=1)

Ta sẽ cấu hình ở phần khai báo như sau.

IER |= M\_INT3;

PieCtrlRegs.PIEIER3.bit.INTx3 = 1;

**Tổng kết lại ta sẽ có cấu hình cho PWM như sau**

Yêu cầu:

Tần số PWM 10Khz.

Băm 2 kênh ePWM3A, ePWM3B.

Thực hiện ngắt PWM và xử lý trong hàm ngắt PWM.

//-------------------------------------------------------

// khai báo

**interrupt** **void** **epwm3\_isr**(**void**);

**void** **InitEPwm3Example**(**void**);

Uint32 EPWM3IntCount;

//---------------------------------------

// trong hàm main

SysCtrlRegs.PCLKCR1.bit.EPWM3ENCLK = 1; // Cấp clock cho EPWM3

GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1; // 0=GPIO, 1=EPWM3A, 2=Resv, 3=Resv

GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1; // 0=GPIO, 1=EPWM3B, 2=Resv, 3=ECAP1

EALLOW;

SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0; // Disable TBCLK within the EPWM

EDIS;

InitEPwm3Example();

EALLOW;

SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1; // Enable TBCLK within the EPWM

EDIS;

EALLOW;

EPWM3IntCount = 0;

PieVectTable.EPWM3\_INT = &epwm3\_isr;

IER |= M\_INT3;

PieCtrlRegs.PIEIER3.bit.INTx3 = 1;

EINT; // Enable Global interrupt INTM

ERTM; // Enable Global realtime interrupt DBGM

EDIS;

//---------------------------------------

// hàm xử lý khi có ngắt

**interrupt** **void** **epwm3\_isr**(**void**){

EPWM3IntCount++;

//Viết chương trình trong này

EPwm3Regs.ETCLR.bit.INT = 1; // Enable ePWM2 INTN pulse

PieCtrlRegs.PIEACK.all = PIEACK\_GROUP3;

}

//---------------------------------------

// hàm init ePWM3

**void** **InitEPwm3Example**(){ // sử dụng init mặc định, rồi tính toán và thay đổi các thông số theo chức năng

EPwm3Regs.TBPRD = 3000; // Set timer period

EPwm3Regs.TBPHS.half.TBPHS = 0x0000; // Phase is 0

EPwm3Regs.TBCTR = 0x0000; // Clear counter

// Setup TBCLK

EPwm3Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN; // Count up

EPwm3Regs.TBCTL.bit.PHSEN = TB\_DISABLE; // Disable phase loading

EPwm3Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1; // Clock ratio to SYSCLKOUT

EPwm3Regs.TBCTL.bit.CLKDIV = TB\_DIV1; // Slow just to observe on the scope

// Setup compare

EPwm3Regs.CMPA.half.CMPA = 1500;

// Set actions

EPwm3Regs.AQCTLA.bit.PRD = AQ\_CLEAR;

EPwm3Regs.AQCTLA.bit.CAU = AQ\_SET;

EPwm3Regs.AQCTLB.bit.PRD = AQ\_CLEAR;

EPwm3Regs.AQCTLB.bit.CBU = AQ\_SET;

// Enable Timer interrupt

EALLOW;

EPwm3Regs.ETSEL.bit.INTSEL = ET\_CTR\_ZERO;// Enable INT on Zero event

EPwm3Regs.ETSEL.bit.INTEN = 1; // Enable INT

EPwm3Regs.ETPS.bit.INTPRD = ET\_1ST;// Generate INT on 1rd event

// set ET\_2TH, ET\_3TH để sau 2,3 chu kỳ ngắt PWM thì nhảy vào hàm xử lý ngắt PWM, tương tự với set ET\_1ST là nhảy vào ngay

EDIS;

}

Để set xung PWM cho hai kênh ePWM3A và ePWM3B ta làm như sau

EPwm3Regs.CMPA.half.CMPA = 1500; // adjust duty for output EPWM3A

EPwm3Regs.CMPB = 2000; // adjust duty for output EPWM3B

Để có thể chạy được chức năng phase shift()(giữa hai kênh PWM với nhau), ví dụ khi thay đổi Epwm4Regs.TBPHS.half.TBPHS thì sẽ có sự lệch pha của kênh PWM4 với kênh PWM tham chiếu

Ta đặt một kênh PWM làm Master (ví dụ PWM3), ta sẽ lấy kênh PWM3 làm kênh PWM gốc, kênh PWM4 khi chạy sẽ lệch phase so với kênh PWM3

Ta cấu hình như sau:

EPwm3Regs.TBCTL.bit.PHSEN = TB\_DISABLE; // MASTER

EPwm3Regs.TBCTL.bit.SYNCOSEL = TB\_CTR\_ZERO; // sync "down-stream"

EPwm4Regs.TBCTL.bit.PHSEN = TB\_ENABLE; // SLAVER

EPwm4Regs.TBCTL.bit.SYNCOSEL = TB\_SYNC\_IN; // Synchronized with SYNC IN signal (either previous ePWM or SWFSYNC for ePWM1)

Sau khi cấu hình như này, để có thể tạo độ lệch pha giữa kênh PWM4 so với PWM3 thì ta chỉ cần set EPwm4Regs.TBPHS.half.TBPHS=PhaseValue;//(giá trị PhaseValue nằm trong 0- TBPRD, tương ứng với lệch 0°-360°)

Cấu hình deadband

EPwm3Regs.DBCTL.bit.IN\_MODE = DBB\_RED\_DBA\_FED;//DBA\_RED\_DBB\_FED;

EPwm3Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE; Enable Dead-band module

EPwm3Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC;//chú ý thanh ghi này

EPwm3Regs.DBRED = 24; //

EPwm3Regs.DBFED = 24; //

**Các chế độ băm xung PWM**

Tùy vào mục đích công việc khác nhau mà ta sẽ băm xung các kiểu khác nhau.

Các biến trong thanh ghi AQCTLA và AQCTLB

// trong file DSP2802x\_EPWM.h

// Action qualifier register bit definitions \*/

struct AQCTL\_BITS { // bits description

Uint16 ZRO:2; // 1:0 Action Counter = Zero

Uint16 PRD:2; // 3:2 Action Counter = Period

Uint16 CAU:2; // 5:4 Action Counter = Compare A up

Uint16 CAD:2; // 7:6 Action Counter = Compare A down

Uint16 CBU:2; // 9:8 Action Counter = Compare B up

Uint16 CBD:2; // 11:10 Action Counter = Compare B down

Uint16 rsvd:4; // 15:12 reserved

};

Các chế độ của các biến

// trong file F2802x\_EPwm\_defines.h

// AQCTLA and AQCTLB (Action Qualifier Control)

//=============================================

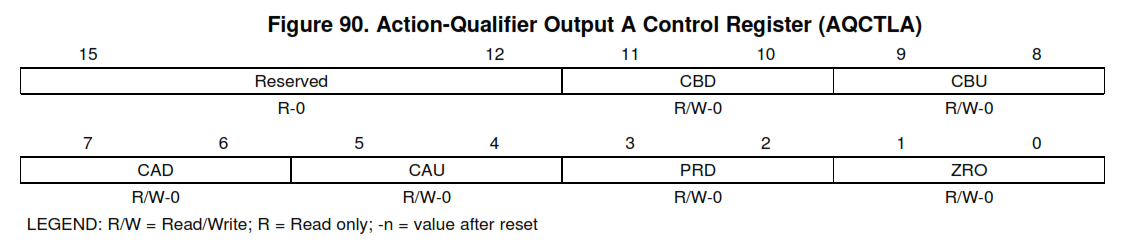
// ZRO, PRD, CAU, CAD, CBU, CBD bits

#define AQ\_NO\_ACTION 0x0

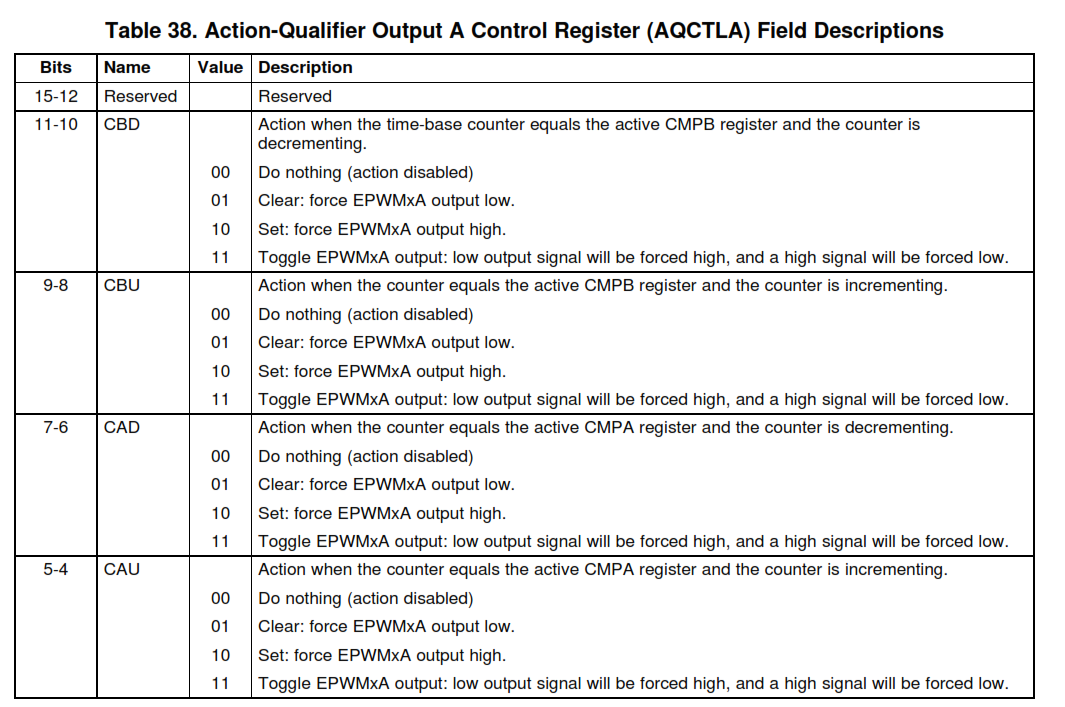
#define AQ\_CLEAR 0x1

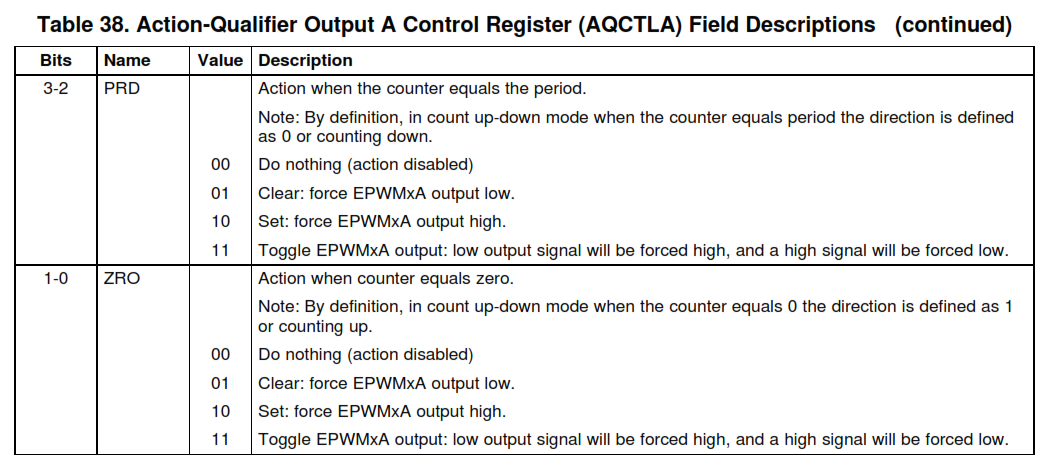
#define AQ\_SET 0x2

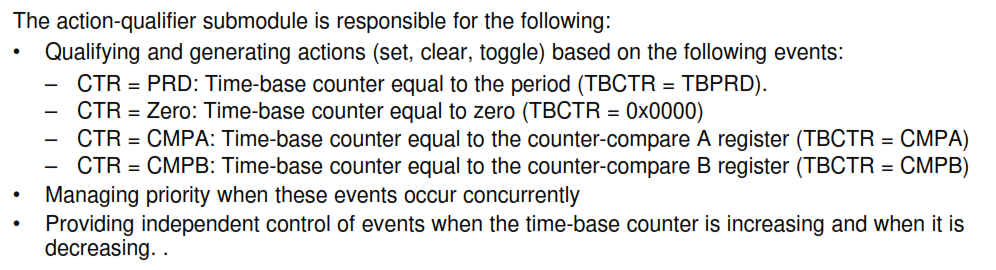
#define AQ\_TOGGLE 0x3



Các action trong thanh ghi AQCTLA







Xét đoạn code sau

EPwm1Regs.AQCTLA.bit.ZRO = AQ\_SET;

EPwm1Regs.AQCTLA.bit.CAU = AQ\_CLEAR;

EPwm1Regs.AQCTLB.bit.ZRO = AQ\_SET;

EPwm1Regs.AQCTLB.bit.CBU = AQ\_CLEAR;

Dựa vào bảng action ta có

EPwm1Regs.AQCTLA.bit.ZRO = AQ\_SET;

EPWM1A high khi Counter = 0;// điểm ban đầu xung PWM

EPwm1Regs.AQCTLA.bit.CAU = AQ\_CLEAR;

EPWM1A low khi counter=CMPA khi tăng

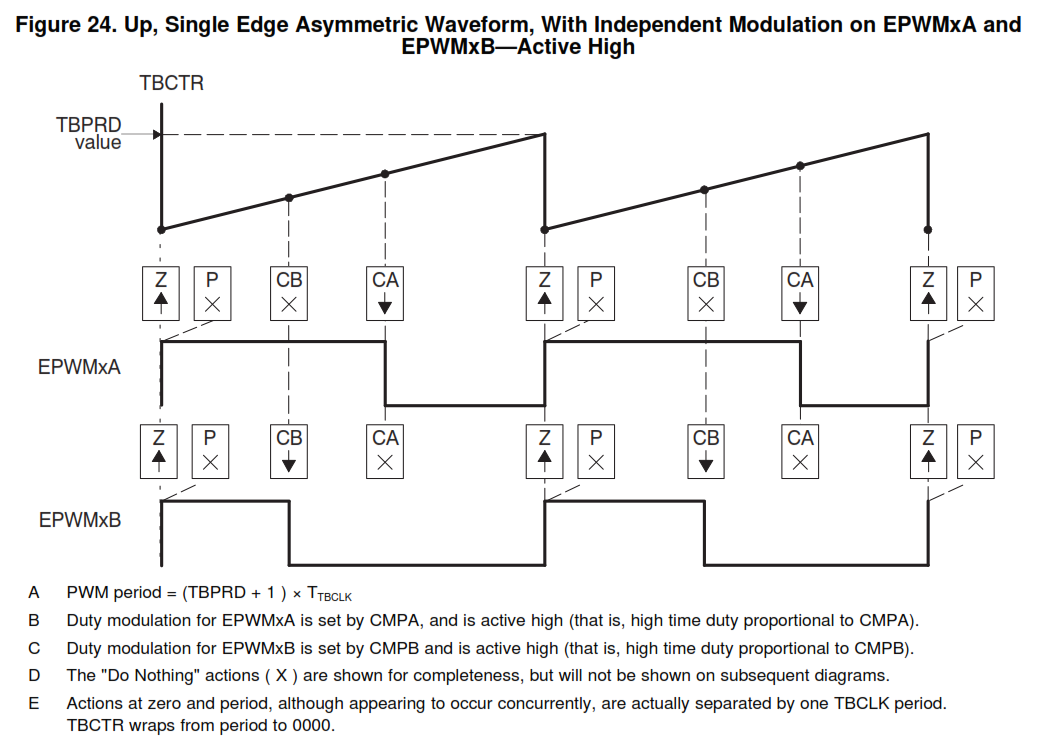
EPwm1Regs.AQCTLB.bit.ZRO = AQ\_SET;

EPWM1B high khi Counter =0;

EPwm1Regs.AQCTLB.bit.CBU = AQ\_CLEAR;

EPWM1B low khi counter=CMPB khi tăng

Kết quả sẽ được hiển thị thông qua bảng dưới.



Tùy vào từng ứng dụng mà sẽ cấu hình thanh ghi AQCTLA/ AQCTLB và một số thanh ghi khác phù hợp.

(sử dụng tài liệu tham khảo đã ghi ở đầu)

1. **Tạo project**

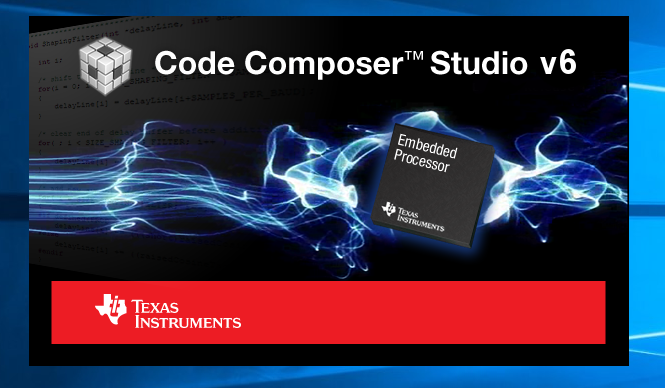
Có nhiều cách để tạo project

Tạo project mới(cách này khá khó cho người mới bắt đầu, do phải cấu hình thư viện phức tạp)

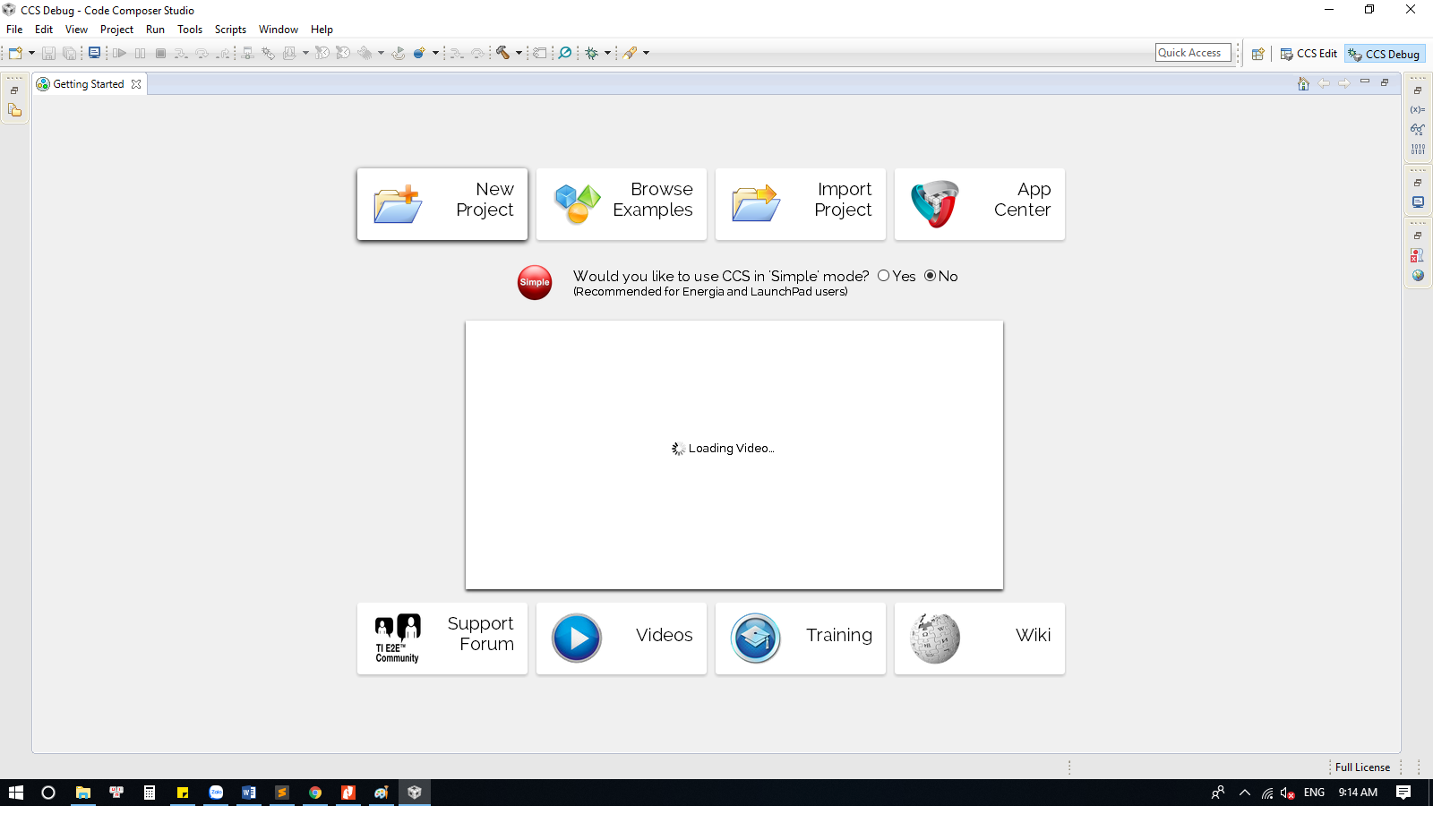
Sử dụng project đã có sẵn rồi export ra project của mình(các thư viện đã được cấu hình, chỉ cần sử dụng lại, cấu hình lại các thanh ghi theo mục đích sử dụng là có thể chạy được).

1. Trong này sẽ hướng dẫn sử dụng cách 2 trước(tạo project từ project đã có sẵn)

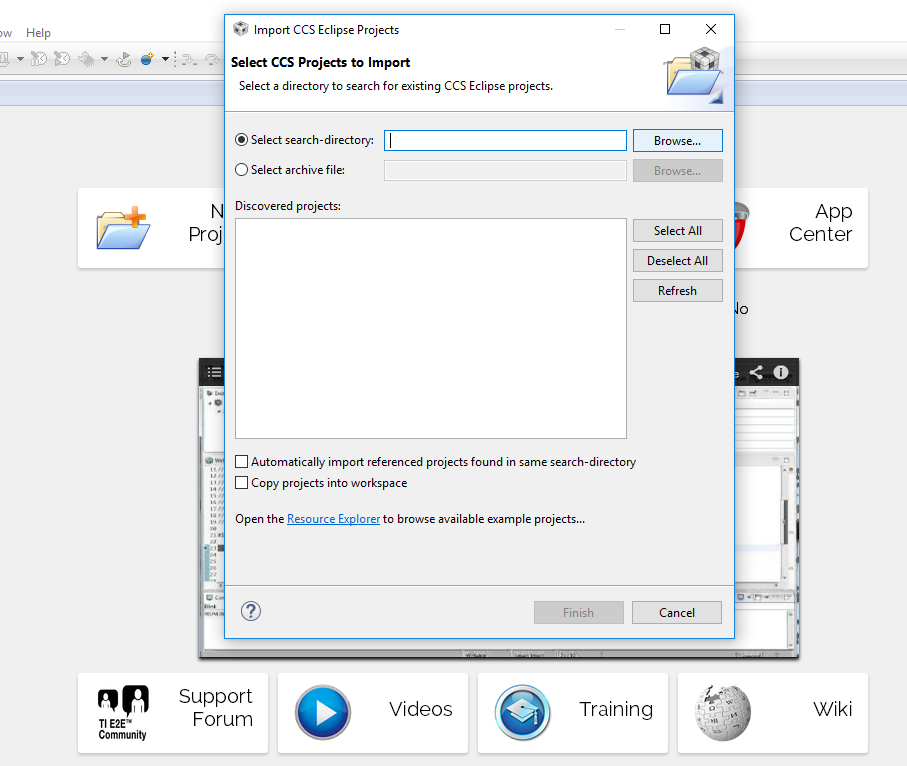
Đầu tiên, mở Code Composer Studio lên .



Từ màn hình chính chọn Import Project.

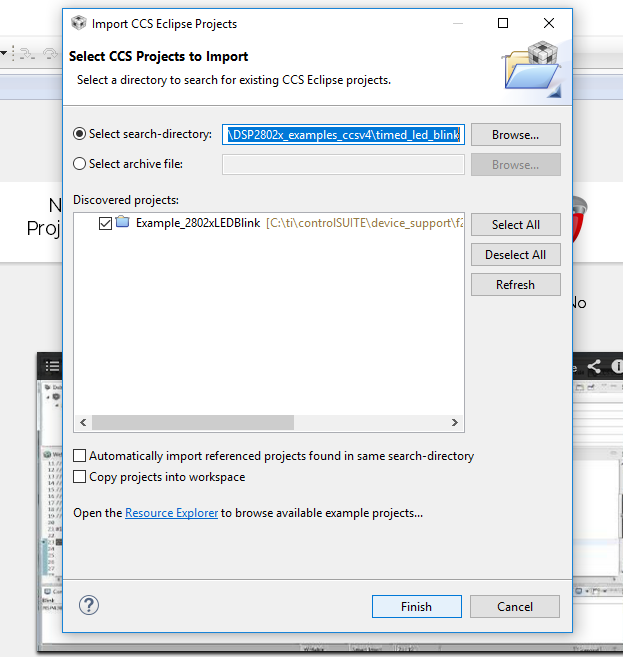


Chọn Browse.

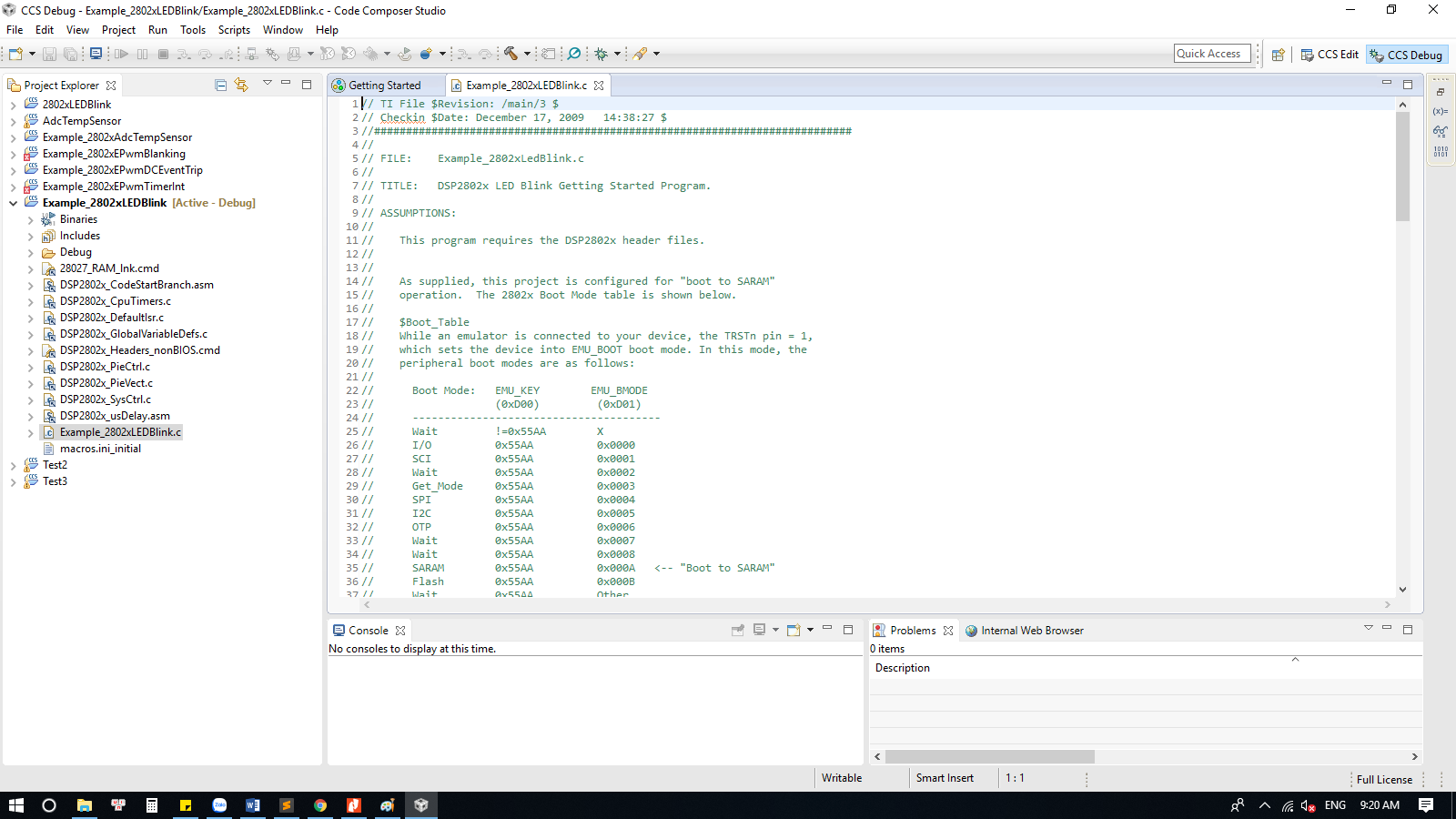


Ta sẽ lấy 1 project bất kỳ(ví dụ như trong link dưới).

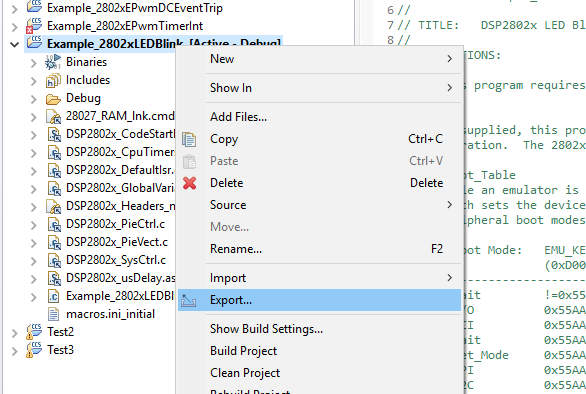
C:\ti\controlSUITE\device\_support\f2802x\v125\DSP2802x\_examples\_ccsv4\timed\_led\_blink



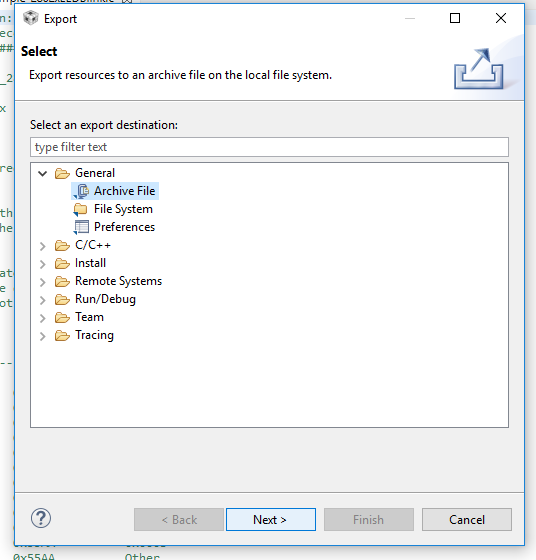
Ta có project mới được mở lên như sau.



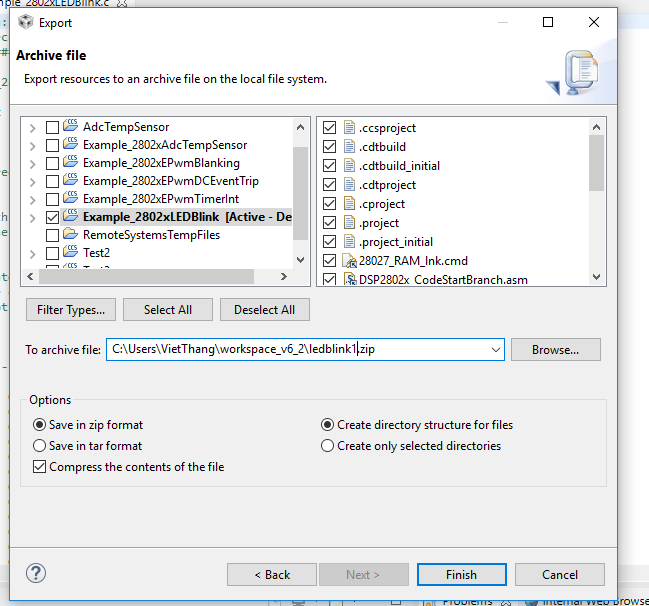
Chọn export.



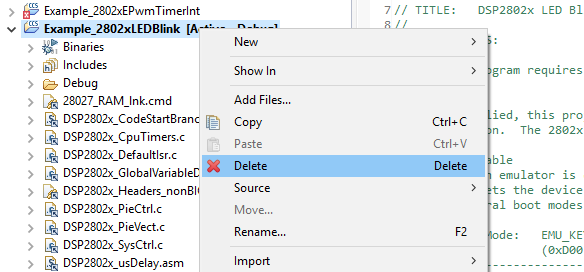
Chọn Archive file rồi next.



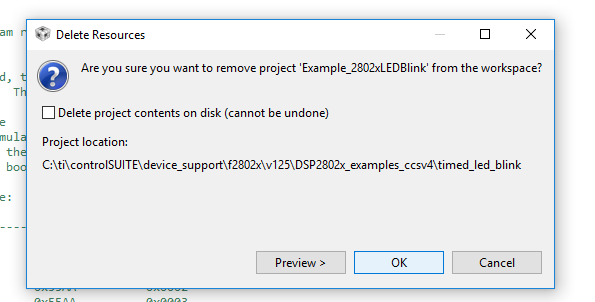
Chọn project muốn export, chọn đường dẫn muốn lưu và định dạng file, sau đó ấn finish.



Đóng project vừa mở(chọn delete).

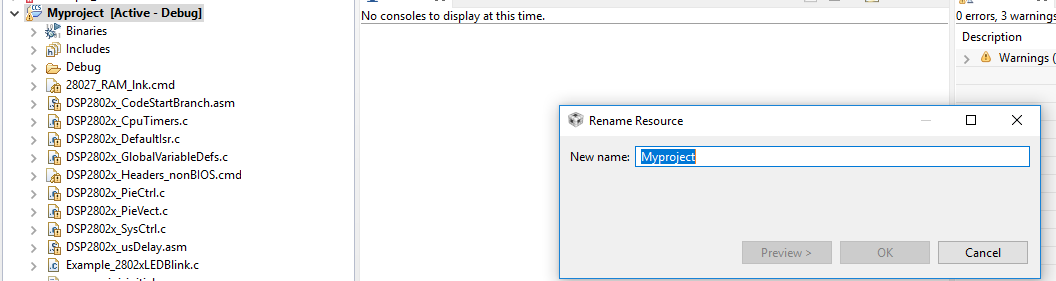


ấn Ok (chú ý ko được click chọn vào ô delete project contens on disk, nếu không sẽ xóa project ở thư mục gốc).

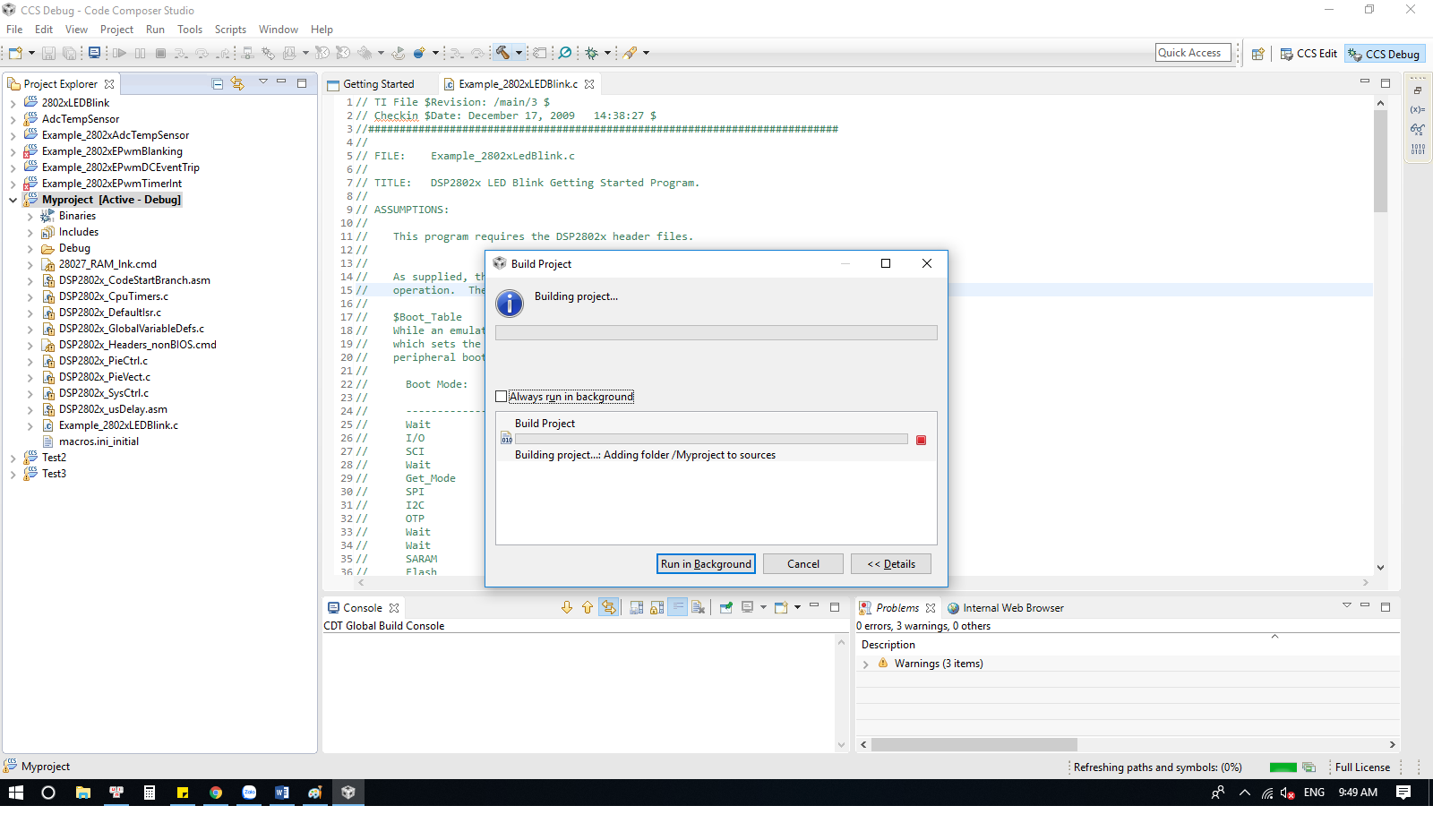


Tiến hành giải nén file vừa export và import lại project đó.

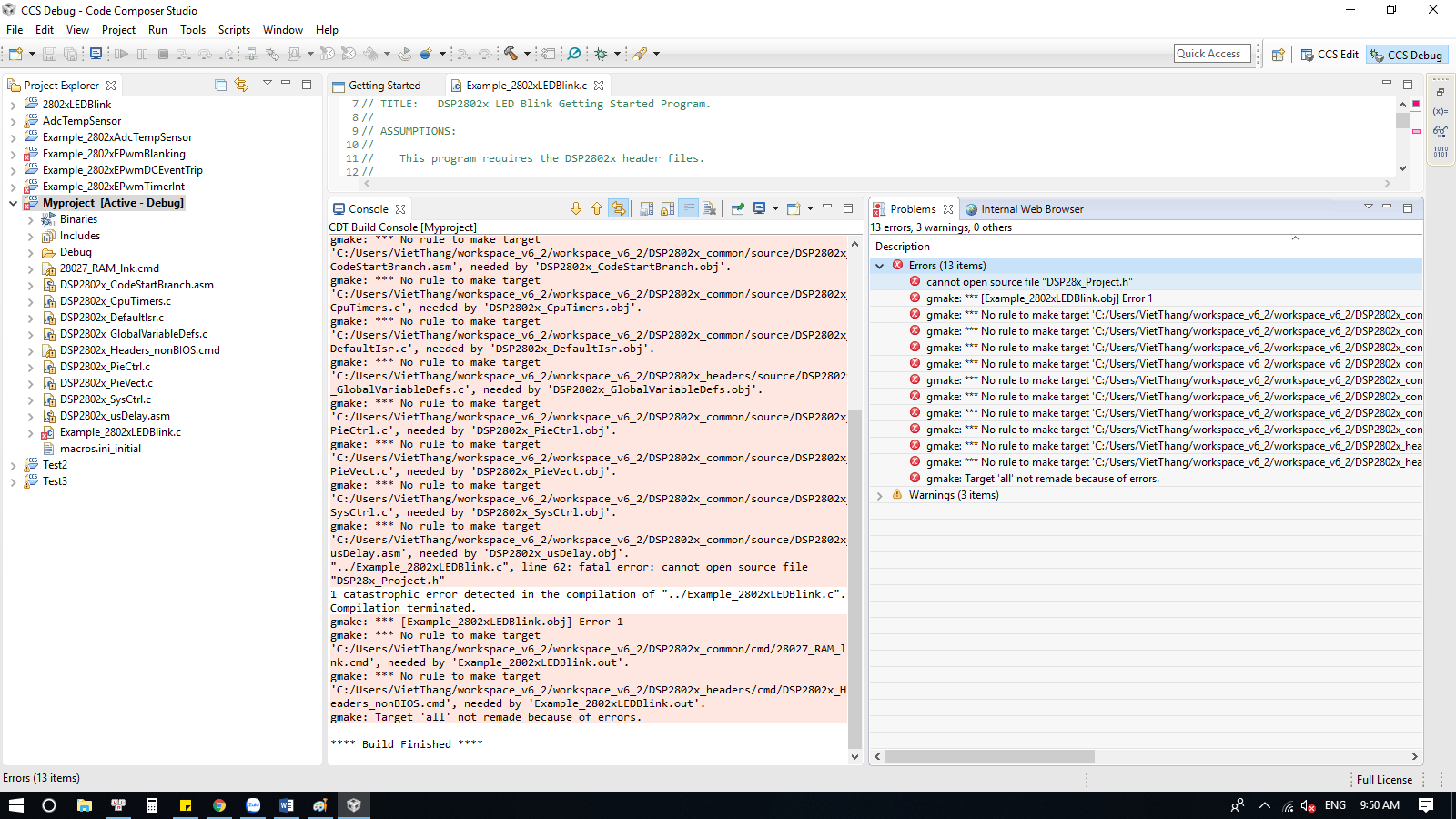
Sau khi import xong thì tiến hành đổi tên project theo mục đích sử dụng.



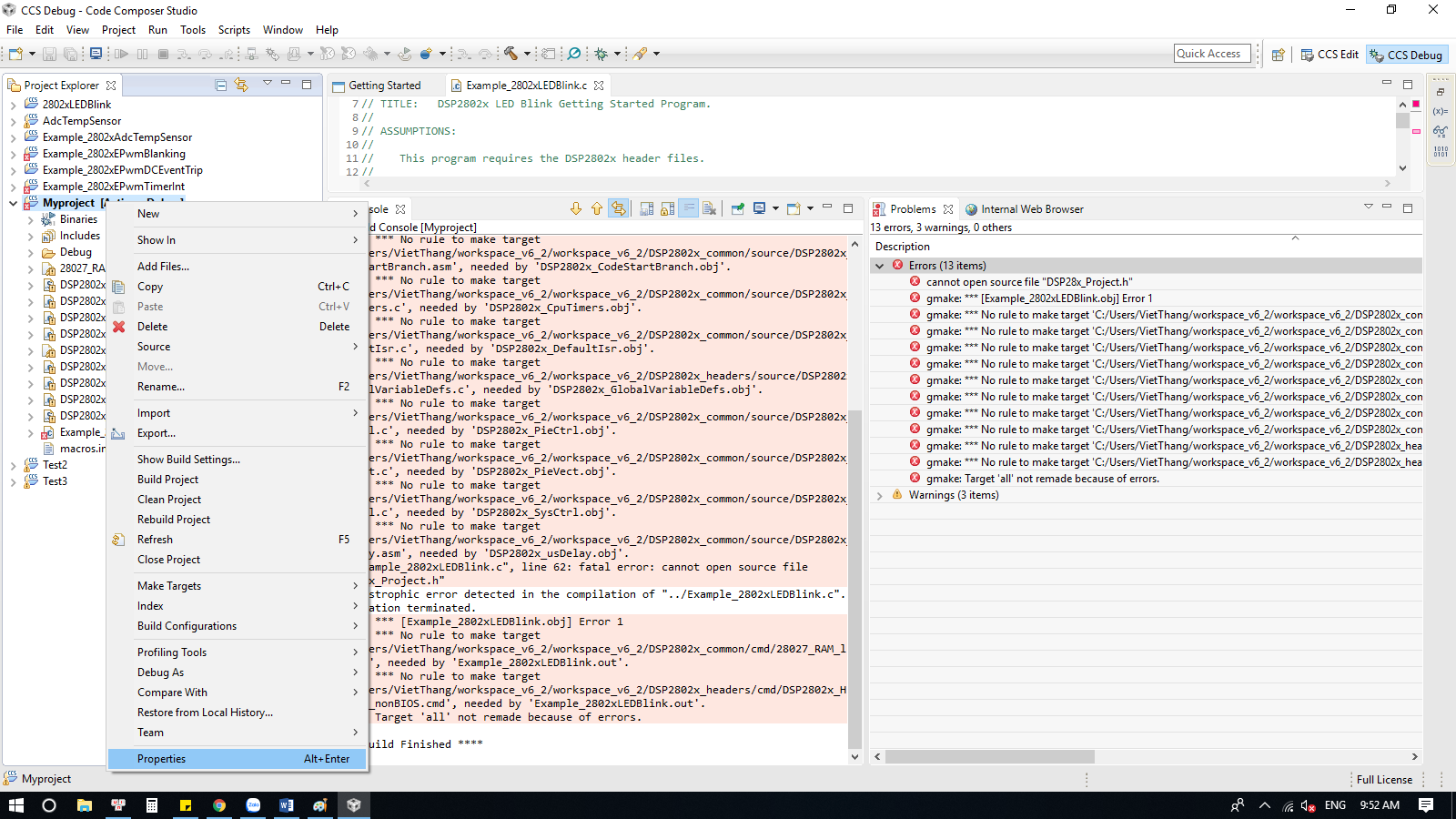
Tiến hành build để xem có lỗi gì thì sửa.



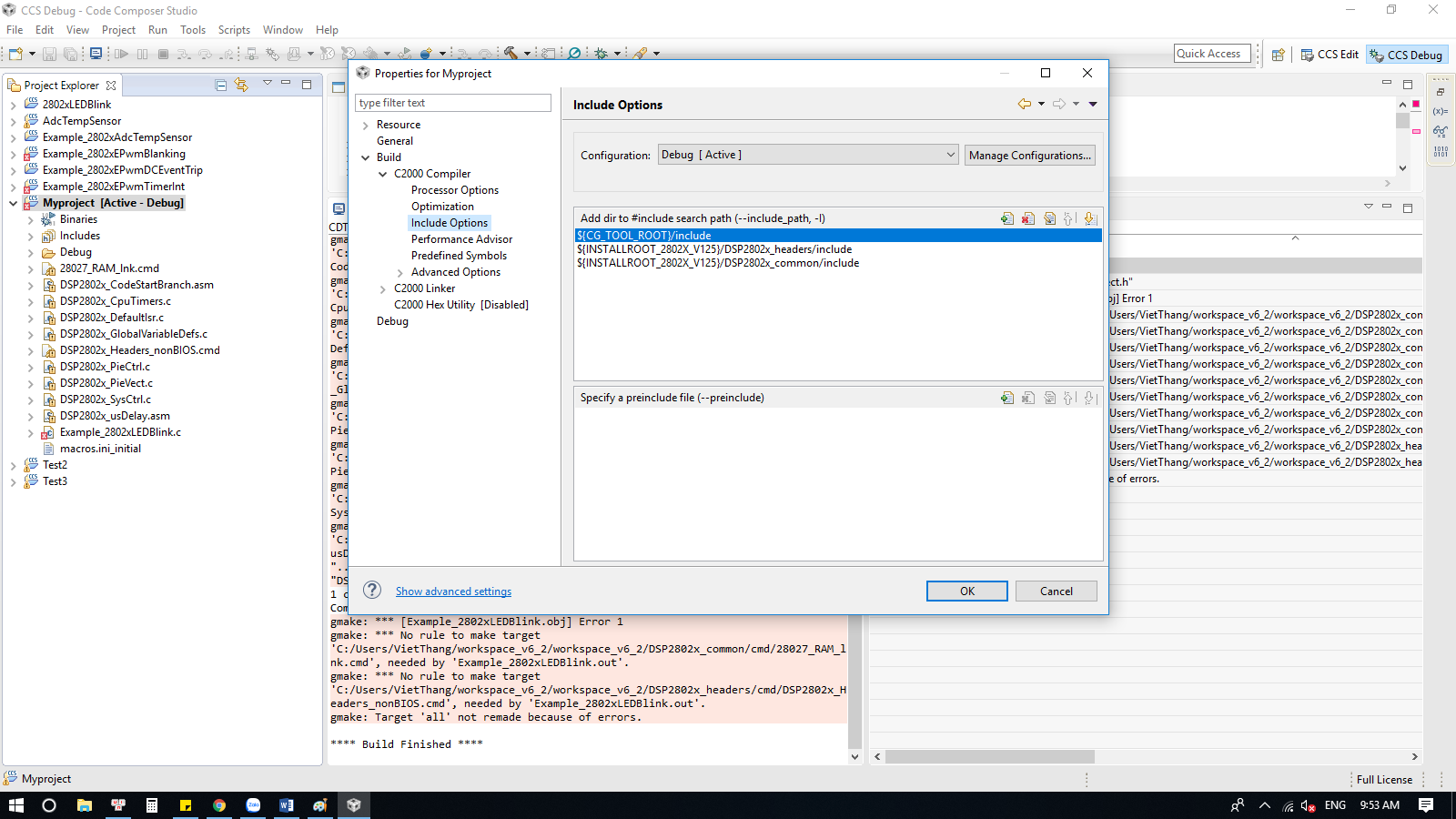
Khi build project mới export thì đường dẫn đến thư viện có sự thay đổi, nên build sẽ bị lỗi.



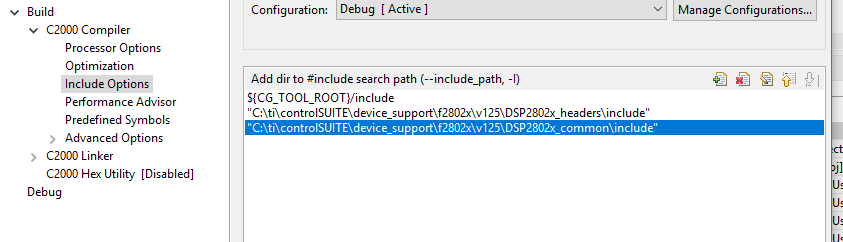
Ta tiến hành thay đổi lại đường dẫn đến thư viện của project gốc.



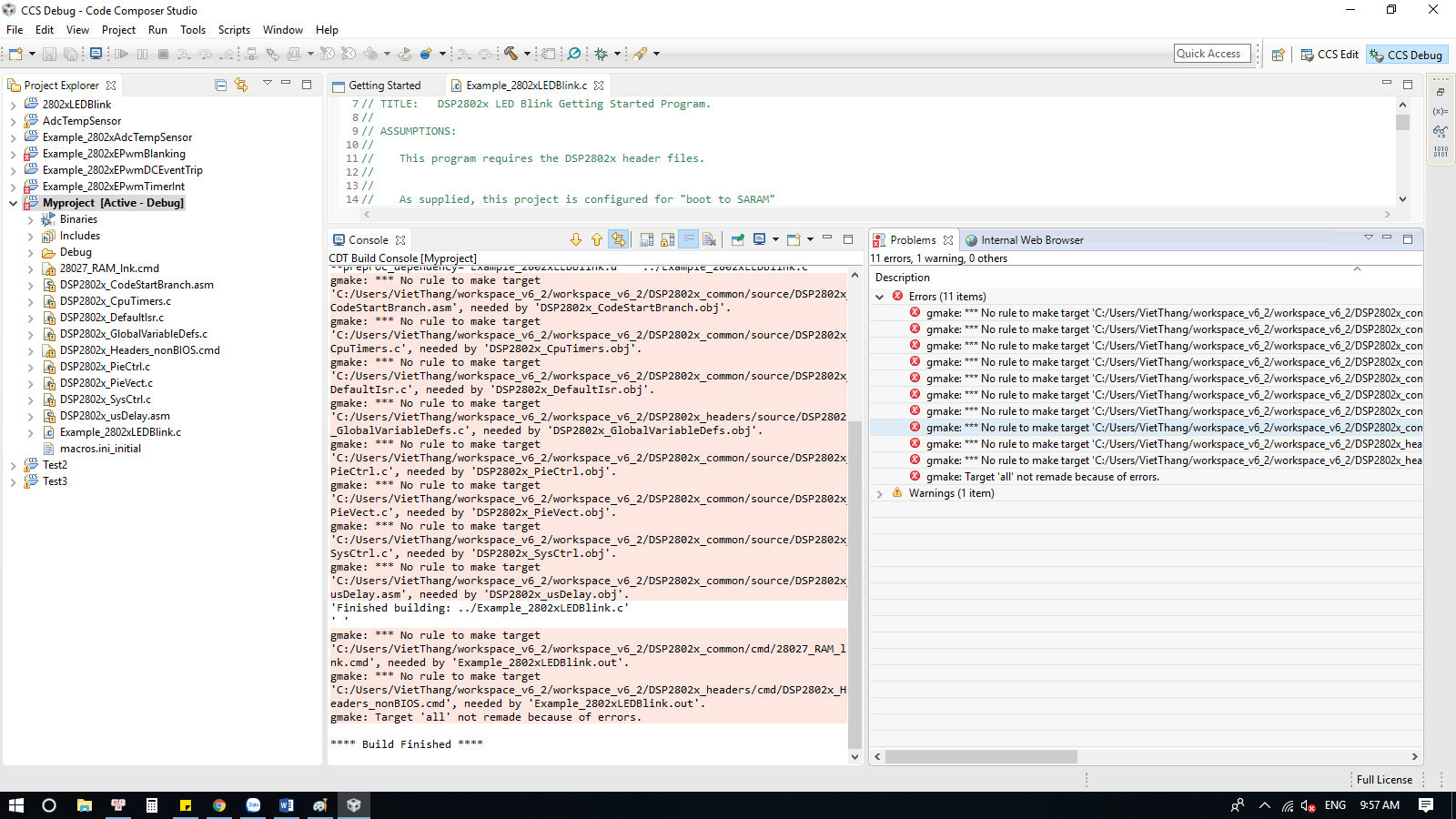
Chọn vào Build->Include Options



Đường dẫn của ${INSTALLROOT\_2802X\_V125}/DSP2802x\_headers/include và ${INSTALLROOT\_2802X\_V125}/DSP2802x\_common/include đang chưa đúng nên ta phải sửa lại như sau : "C:\ti\controlSUITE\device\_support\f2802x\v125\DSP2802x\_headers\include" và "C:\ti\controlSUITE\device\_support\f2802x\v125\DSP2802x\_common\include"

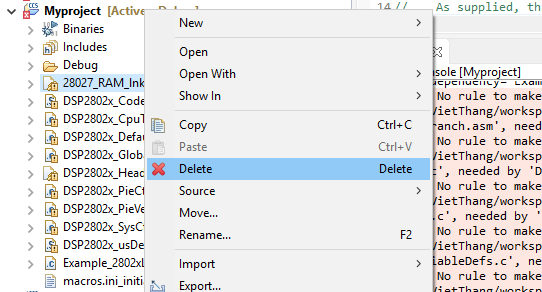


Xong rồi thoát ra và build lại.



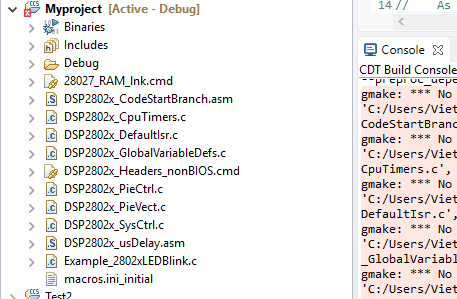
Lỗi này là do link tới các thư viện trong thư mục dự án chưa đúng nên sửa lại như sau

Chọn vào các file có dấu ! và chọn delete, sau đó chọn ok.

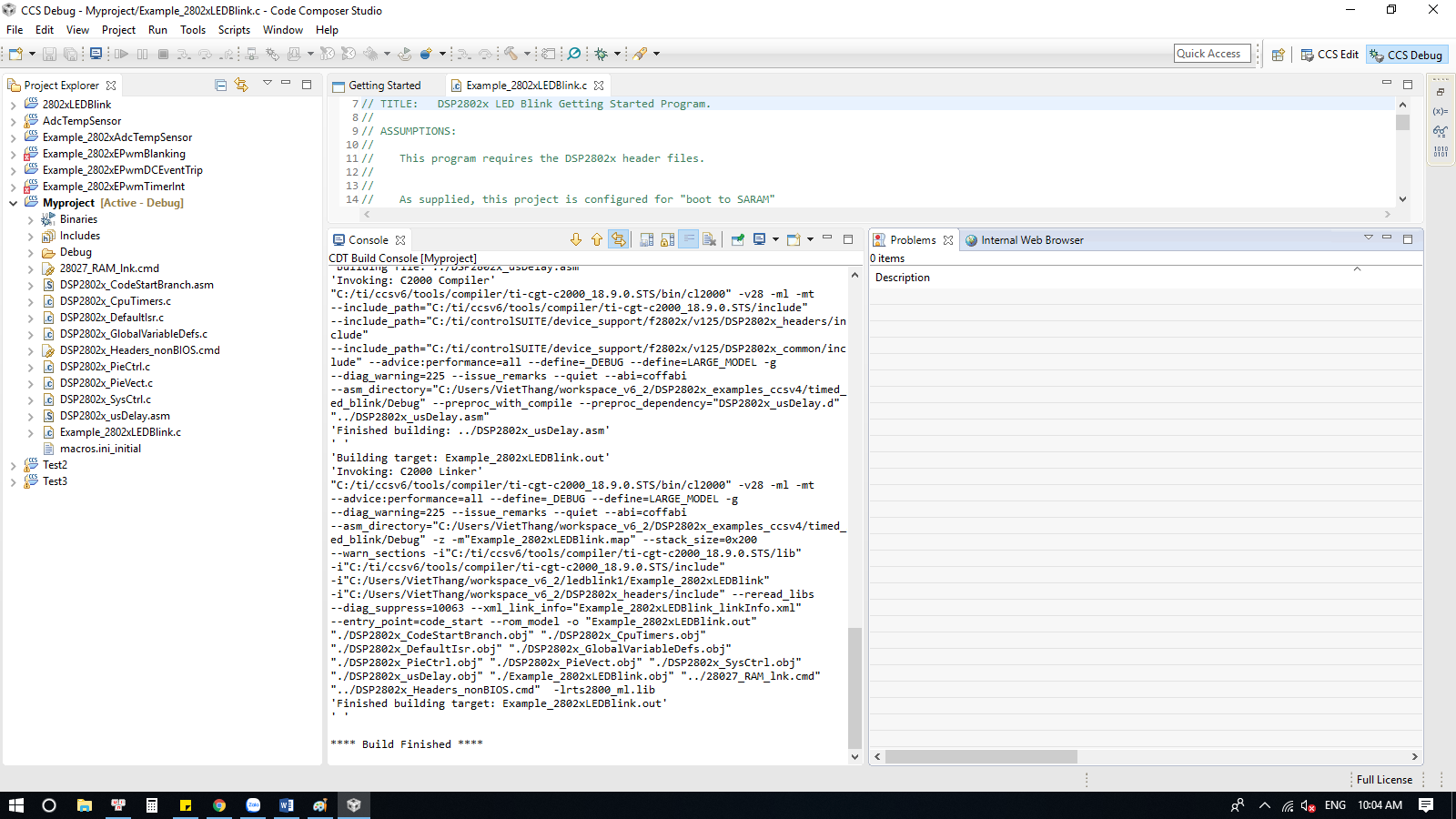


Khi export project thì các file này được copy vào project, nhưng đường dẫn các file này đến thư mục gốc vẫn giữ nguyên nên bị sai do đó chỉ cần xóa các đường dẫn để trình biên dịch gọi đúng là được.

Kết quả như sau.

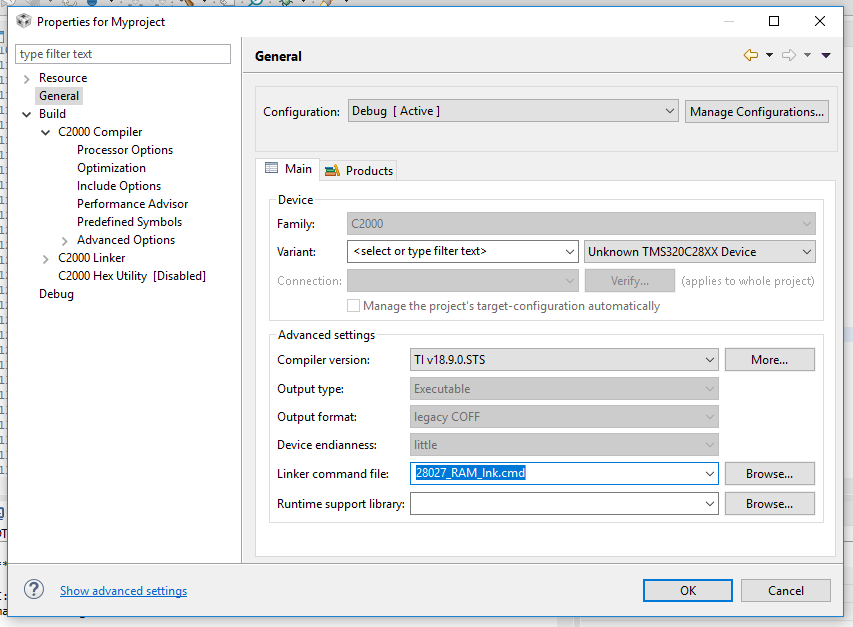


Tiến hành build lại.



Build đã thành công, không có lỗi khi biên dịch, nhưng để có thể nạp vào CPU và có thể chạy được thì phải thêm một số bước nữa.

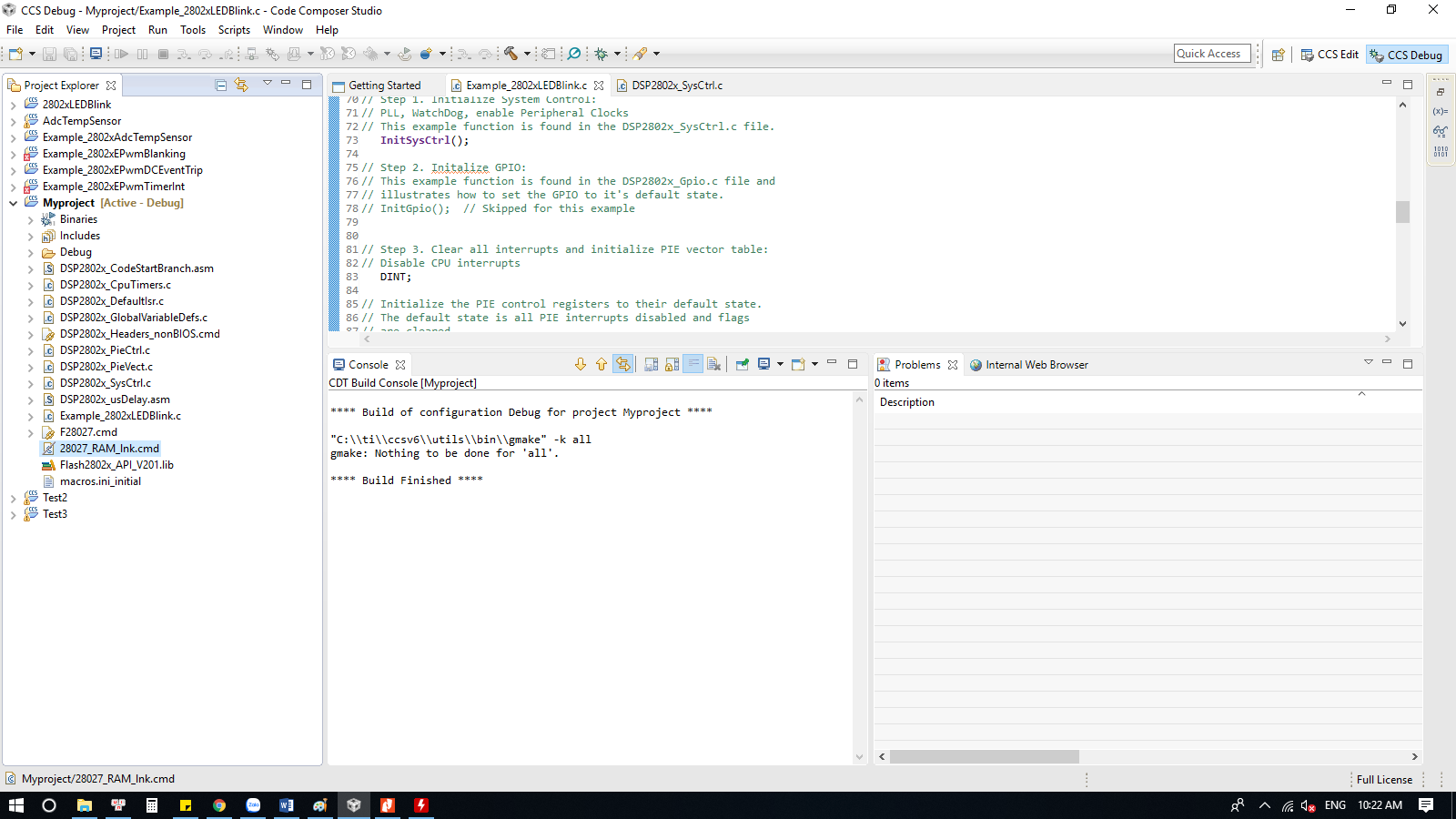
Chọn properties->General->Linker command file, xóa đi để trống và chọn ok.



Tìm file F28027.cmd, Flash2802x\_API\_V201.lib và copy vào thư mục project.

Các file này có thể tìm trong thư mục C:\ti\controlSUITE

Tiến hành build lại.



Lúc này đã có thể cho CPU hoạt động và ta có thể tiến hành cấu hình theo mục đích sử dụng và viết chương trình để chạy (có thể tham khảo các tài liệu đã có), việc thêm chức năng mới có thể add thêm thư viện khác để chạy.

Lúc này có thể tạo project mới và define lại giống như trong project này là có thể chạy được.