

Mục Lục

I. Đặt vấn đề :	4
II. Phần mềm biên dịch Psoc Designer.	7
1. Tổng quan	7
2. Sử dụng	7
a. Khởi động chương trình Psoc Designer:	7
b. Thiết lập cấu hình	9
c. Soạn thảo chương trình	17
d. Tra cứu datasheet của các modul	18
III. Mạch nạp và phần mềm nạp chương trình.	20
1. Mạch nạp	20
2. Hướng dẫn sử dụng mạch nạp PSOC và chương trình nạp	21
IV. Các bài thực hành.	22
Bài 1: Led đơn	22
1. Mạch nguyên lý	22
2. Kết nối cáp	23
3. Viết chương trình	23
Bài 2. Led 7 thanh	26
1. Sơ đồ mạch nguyên lý	26
2. Nối cáp	27
3. Viết chương trình	27
Bài 3. Ma trận led	29
1. Cấu tạo ma trận led	29
2. Sơ đồ nguyên lý	29
3. Mạch trên kit	30
4. Nối cáp	31
5. Viết chương trình	31
Bài 4. Ma trận bàn phím	33
1. Sơ đồ nguyên lý	33
2. Mạch trên Kit	34
3. Nối cáp	34
4. Chương trình	34
Bài 5. LCD	36
1. Sơ đồ nguyên lý	36
2. Mạch trên kit	37
3. Nối cáp	37

4. Code chương trình.....	37
Bài 6. Mạch cầu H - PWM.....	39
1. Sơ đồ nguyên lý.....	39
2. Mạch trên kit.....	40
3. Nối cáp.....	41
4. Code chương trình.....	41
Bài 7. ADC.....	47
1. Sơ đồ nguyên lý.....	47
2. Mạch trên kit.....	48
3. Nối cáp.....	48
4. Code chương trình.....	48
Bài 8. LM35.....	53
1. Sơ đồ nguyên lý.....	53
2. Mạch trên kit.....	53
3. Nối cáp.....	53
4. Code chương trình.....	54
Bài 9. Truyền thông Uart.....	55
1. Sơ đồ nguyên lý.....	55
2. Mạch trên kit.....	55
3. Nối cáp.....	56
4. Code chương trình.....	56
Bài 10. DS1307.....	60
1. Sơ đồ nguyên lý.....	60
2. Mạch trên kit.....	60
3. Nối cáp.....	61
4. Code chương trình.....	61
Bài 11. Timer.....	65
1. Sơ đồ nguyên lý.....	65
2. Đấu nối dây.....	66
3. Code chương trình.....	69
Bài 12. Đo tốc độ động cơ (Timer, Counter).....	70
1. Sơ đồ nguyên lý.....	70
2. Nối cáp.....	70
3. Code chương trình.....	70
Bài 13. Ngắt GPIO.....	74
1. Sơ đồ nguyên lý.....	74
2. Thiết lập cấu hình.....	74

3. Code chương trình.....	76
Bài 14. Ngắt GPIO trên 2 chân bất kỳ.....	77
1. Sơ đồ nguyên lý.....	77
2. Thiết lập cấu hình.....	77
3. Code chương trình.....	79
V. Kết luận.	80
VI. Tài liệu tham khảo.	80

Lời nói đầu

PSOC là họ vi điều khiển khá mạnh với tốc độ xử lý lên tới 24Mlps, thư viện các ngoại vi phong phú, đủ cho đa số các ứng dụng, khả năng cấu hình mềm dẻo, cấu trúc bao gồm cả các module tương tự và số, cho phép xây dựng các ứng dụng với số lượng ngoại vi ít nhất.

Việc tiếp cận dòng vi điều khiển này gặp nhiều khó khăn do: tài liệu về PSoC không nhiều, đặc biệt là tài liệu bằng tiếng Việt. Đặc biệt là giao diện Design Editor sử dụng nhiều khái niệm mới, không có trong các họ Vi điều khiển khác, gây lung túng cho người mới bắt đầu học. Việc xây dựng phần cứng cũng là trở ngại khi học các dòng vi điều khiển mới.

KIT phát triển PSoC đã được xây dựng cho phép người học có thể nhanh chóng xây dựng các ứng dụng trên cơ sở các dòng PSoC và các ngoại vi cơ bản. Tài liệu này được soạn nhằm giúp người học nhanh chóng tiếp cận họ VDK PSoC và sử dụng kit phát triển PSoC.

Toàn bộ những công việc trên đã được hoàn thành với sự đóng góp công sức rất lớn của các em sinh viên ngành Trang bị điện các khoá 45, 47 và 48.

Mọi ý kiến đóng góp xin gửi về: Nguyễn Văn Nghĩa - bộ môn Kỹ thuật điện – ĐHGTVT.

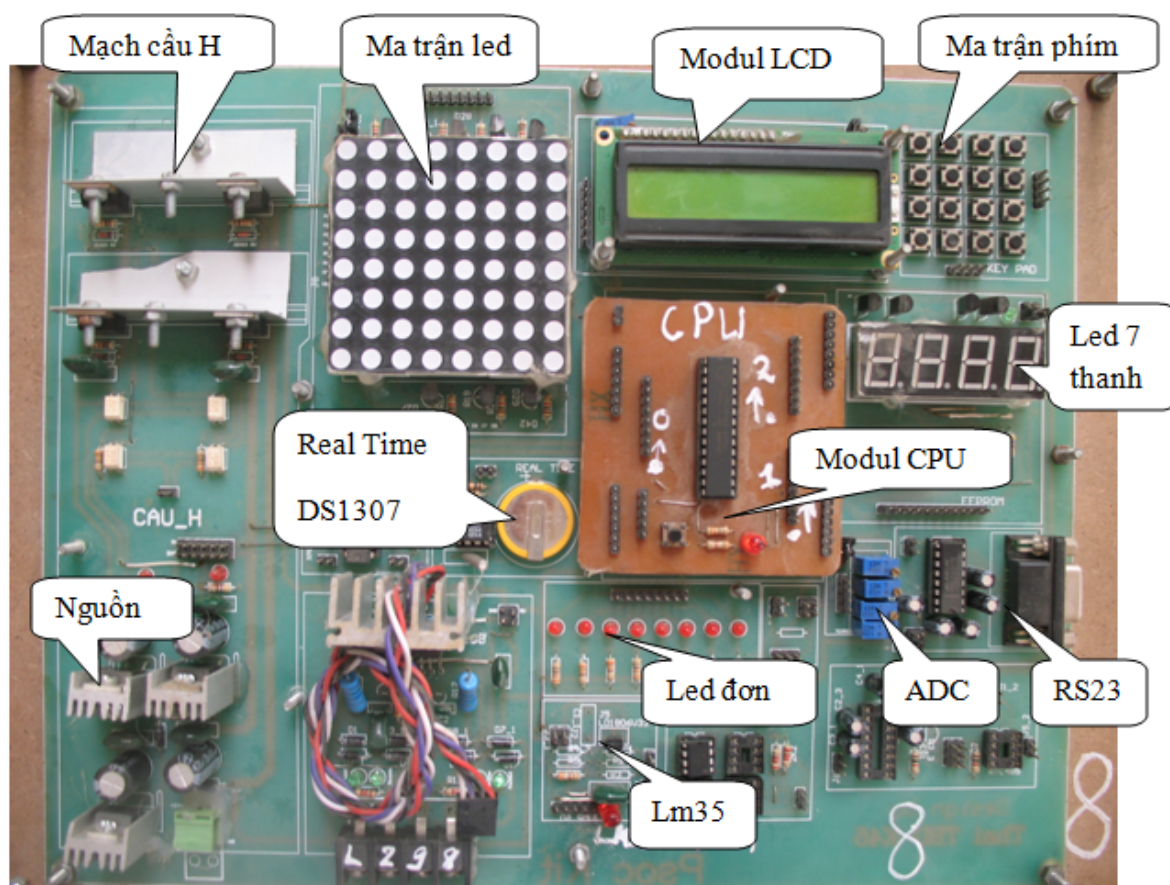
Email: nguyennghia.nh@gmail.com

I. Đặt vấn đề :

Xu hướng gắn lý thuyết trong nhà trường với thực tiễn, gắn những nghiên cứu trong trường học với những vấn đề của xã hội là một nhu cầu tất yếu và phù hợp với quy luật phát triển. Để đào tạo được nhân lực có trình độ cao và có khả năng tiếp cận nhanh chóng với thực tiễn sản xuất thì việc trang bị các thiết bị phục vụ cho việc thực hành, thí nghiệm là vô cùng cần thiết. Tuy nhiên hệ thống các thiết bị thực hành do nước ngoài cung cấp có giá thành rất lớn khó phù hợp với điều kiện thực tế của nhiều trường, cơ sở đào tạo trong nước. Việc làm từng mạch thật cho mỗi bài rất mất thời gian và công sức. Nếu sử dụng kit thực hành thì có thể tiết kiệm được thời gian làm mạch và những sai sót do làm mạch.

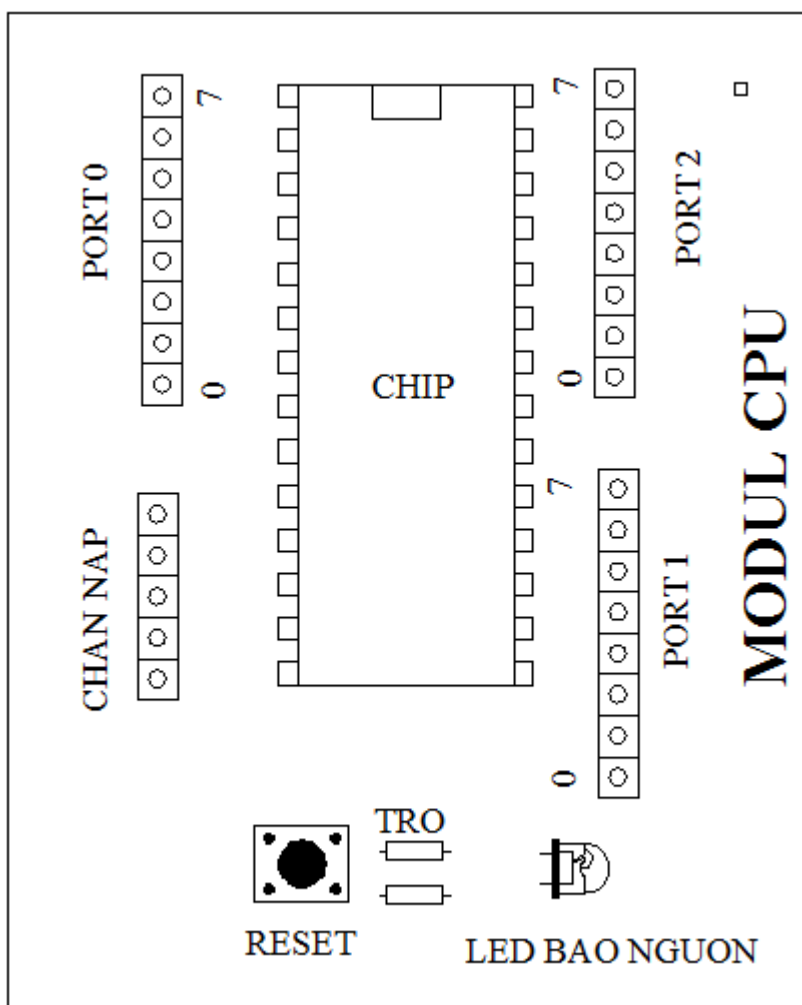
Họ vi điều khiển Psoe hiện đang được sử dụng ngày càng rộng rãi trong thực tế và đã có rất nhiều trường đại học, cao đẳng, trung cấp chuyên nghiệp đưa vào giảng dạy. Việc xây dựng thiết bị thực hành, kit phát triển để có thể học tập đạt hiệu quả cao là nhu cầu rất bức thiết.

Hình ảnh Kit:



Kit thực hành gồm có:

1. Modul nguồn: Cung cấp nguồn cho toàn bộ kit.
2. Modul CPU:

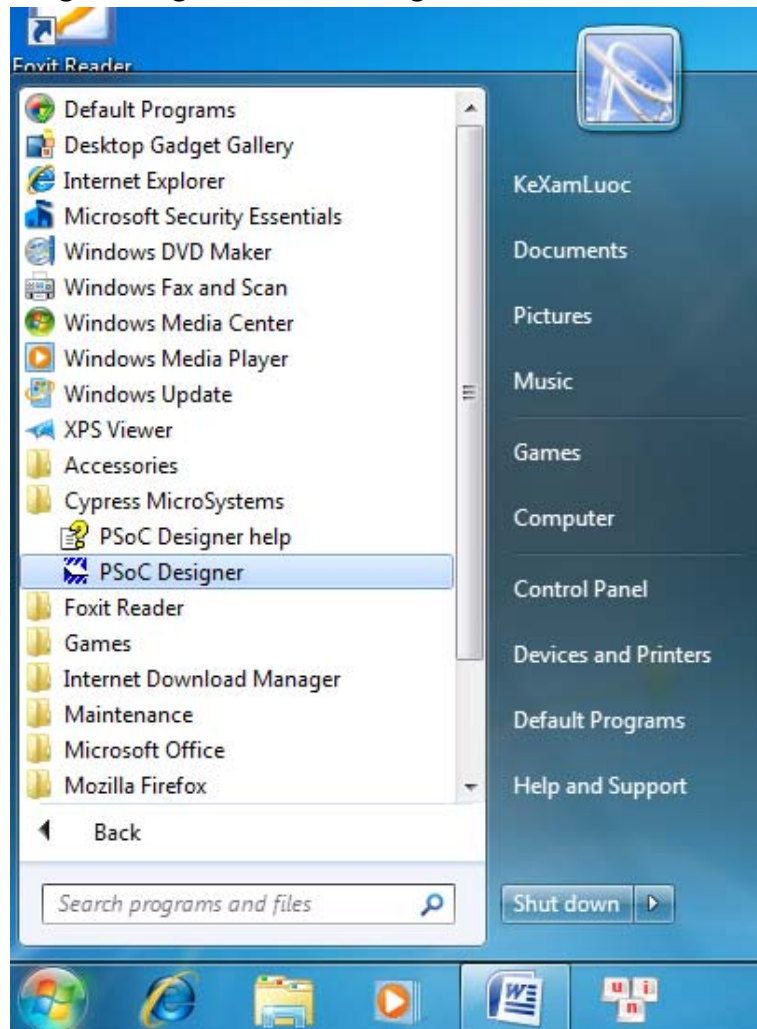


Modul CPU gồm:

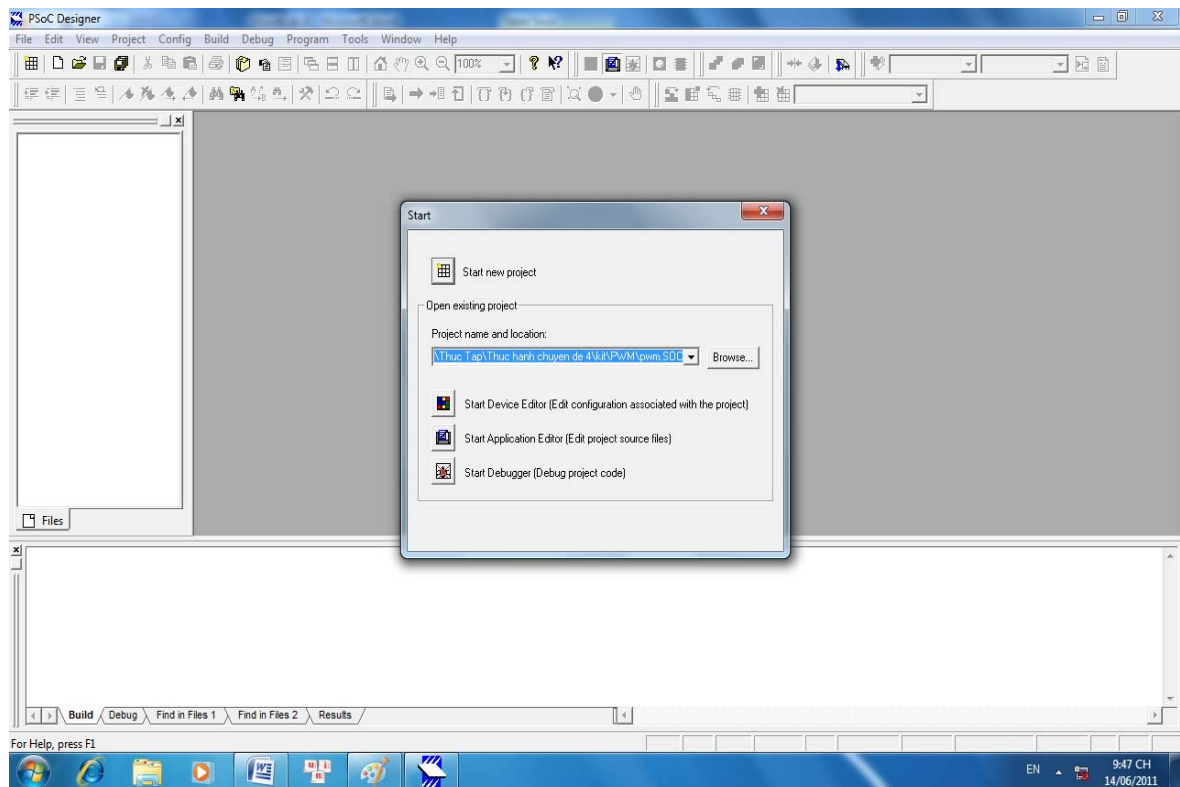
- Chip Psoc 24966.
 - Jum để nối với mạch nạp để nạp chương trình từ máy tính xuống chip.
 - Led báo khi modul được cấp nguồn.
 - Nút bấm reset chip.
 - Các Port của chip.
3. Modul Led đơn.
 4. Modul Led 7 thanh.
 5. Modul ma trận phím.
 6. Modul ma trận led.
 7. Modul ADC.
 8. Modul đo nhiệt độ.
 9. Modul mạch cầu H.
 10. Modul giao tiếp RS232.
 11. Modul giao tiếp DS1307.
 12. Và một số modul khác nữa như Rom, xác định chiều động cơ...

II. Phần mềm biên dịch Psoc Designer.

1. Tổng quan.
2. Sử dụng.
 - a. Khởi động chương trình Psoc Designer:



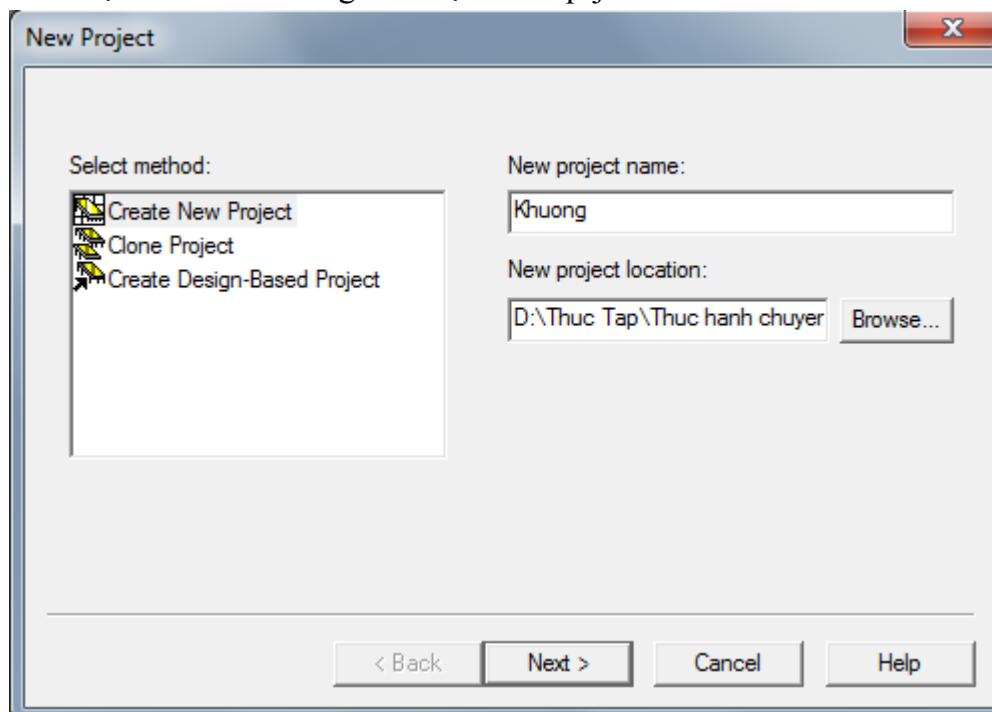
Sau khi khởi động xong ta có giao diện như sau:



Để tạo 1 Project mới ta click vào Start new Project.

Đặt tên cho Project ở mục: New project name.

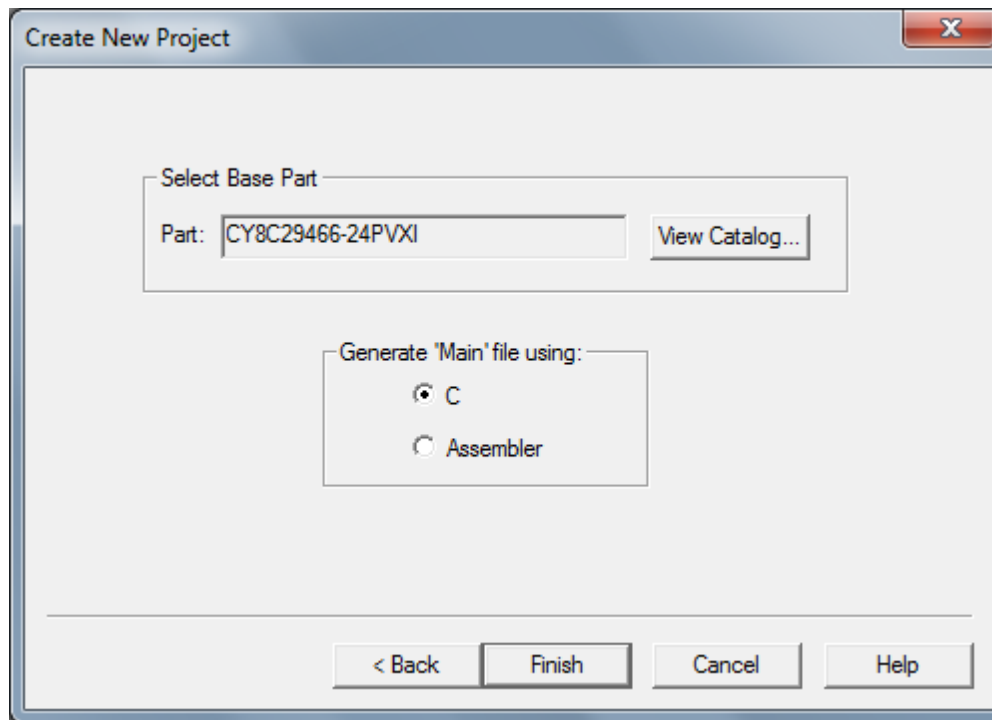
Và chọn nơi lưu chương trình tại: New prjoect location .



Sau khi nhập tên ta chọn NEXT và chọn YES để tạo 1 Project mới.

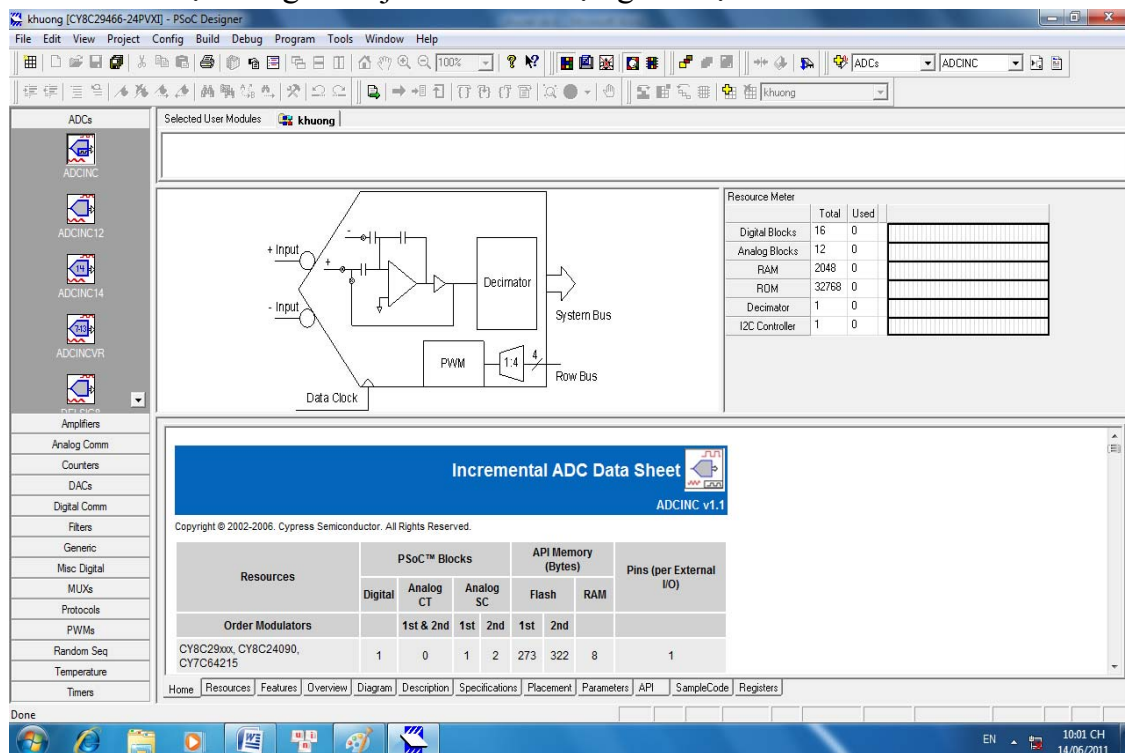
Tiếp theo ta chọn loại chip mà mình sử dụng ở mục: Select Base Part.

Chọn ngôn ngữ lập trình ở mục: Generate 'Main' file using.



Click Finish để kết thúc quá trình tạo Project mới.

Sau khi tạo xong 1 Project mới ta được giao diện như sau:



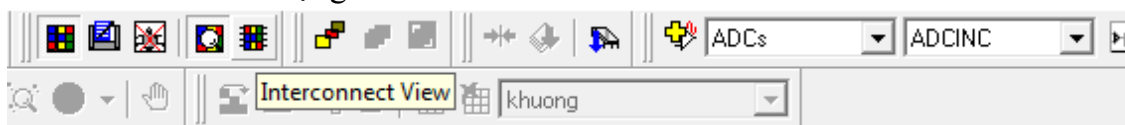
b. Thiết lập cấu hình.

Các Pin của họ PsoC đều là các Pin đa chức năng. Nó có thể sử dụng làm đầu vào, đầu ra... Vì vậy để các chân hoạt động đúng với chức năng mà ta định sử dụng thì cần thiết lập cấu hình cho các chân đây.

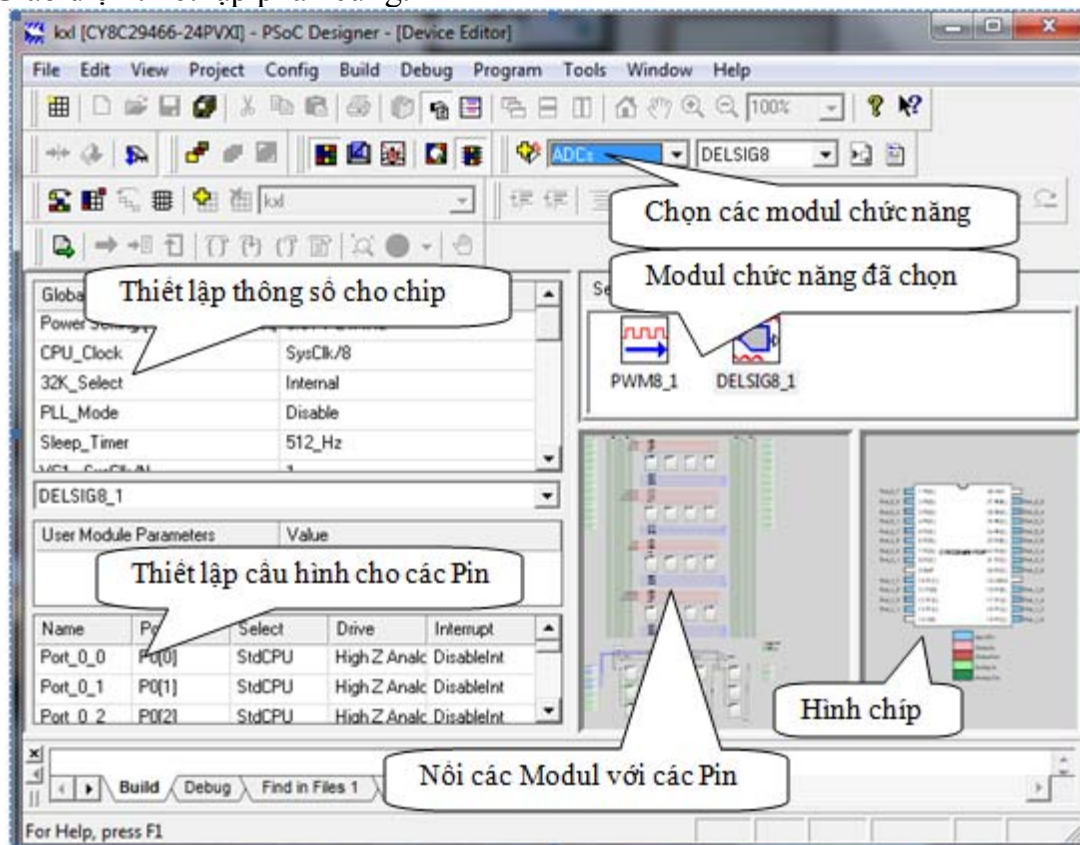
Ngoài ra, không như các dòng vi điều khiển khác các chân ngắt, Pwm, ADC, chân truyền thông.... đều được cấu hình mặc định ở một số chân nhất định. Dòng PsoC thì ta có thể thay đổi được vị trí các chân này một cách tương đối linh hoạt bằng cách cấu hình phần cứng cho nó.

Để thiết lập cấu hình cho các chân, modul tương ứng với mục đích sử dụng

Click vào biểu tượng Interconnect View  trên tab bar.



Giao diện thiết lập phần cứng.



- Thiết lập thông số cho chip hoạt động:

Global Resources	Value
Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	1
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

Power Setting[Vcc/SysClk freq] : Thiết lập điện áp, tần số dao động cho chip hoạt động. Có các mức sau:

- + 3,3V – 24Mhz.
- + 3,3V – 6Mhz.
- + 5 V – 24Mhz.
- + 5 V – 6Mhz.

CPU_clock: Số chu kỳ thạch anh ứng với 1 lệnh.

- + Sysclk/1 : 1 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/2 : 2 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/4 : 4 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/8 : 8 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/16 : 16 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/32 : 32 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/128 : 128 chu kỳ thạch anh tương ứng 1 lệnh.
- + Sysclk/256 : 256 chu kỳ thạch anh tương ứng 1 lệnh.

32k_select: Bộ nhớ trong Internal, Ngoài External.

VC1 = Sys/N : Chia tần số cho các modul chức năng.

- + N nhận giá trị từ 0 đến 8.

VC2 = Sys/N : Chia tần số cho các modul chức năng.

- + N nhận giá trị từ 0 đến 8.

VC3 = Sys/N : Chia tần số cho các modul chức năng.

- + N nhận giá trị từ 0 đến 256

SysClk Source: Nguồn cấp xung clock.

+ Internal: Lấy từ thạch anh bên trong.

+ External: Lấy từ thạch anh ngoài.

Ngoài ra còn 1 số thiết lập khác sẽ đề cập đến sau.

- Thiết lập cấu hình cho các Pin.

Nếu các Pin được nối với 1 modul nào đó (như Pwm, LCD) thì các Pin sẽ tự cấu hình cho tương ứng. Nhưng nếu ta không kết nối với 1 modul nào thì ta sẽ phải cấu hình chúng cho tương ứng với mục đích sử dụng của Pin đấy.

Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	High Z Ana	DisableInt
Port_0_1	P0[1]	StdCPU	High Z	bleInt
Port_0_2	P0[2]	StdCPU	High Z Analog	bleInt
Port_0_3	P0[3]	StdCPU	Open Drain High	bleInt
Port_0_4	P0[4]	StdCPU	Open Drain Low	bleInt
Port_0_5	P0[5]	StdCPU	Pull Down	bleInt
Port_0_6	P0[6]	StdCPU	Pull Up	bleInt
Port_0_7	P0[7]	StdCPU	Strong	bleInt
Port_0_8	P0[8]	StdCPU	Strong Slow	bleInt
Port_1_0	P1[0]	StdCPU	High Z Analog	DisableInt
Port_1_1	P1[1]	StdCPU	High Z Analog	DisableInt
Port_1_2	P1[2]	StdCPU	High Z Analog	DisableInt
Port_1_3	P1[3]	StdCPU	High Z Analog	DisableInt
Port_1_4	P1[4]	StdCPU	High Z Analog	DisableInt
Port_1_5	P1[5]	StdCPU	High Z Analog	DisableInt
Port_1_6	P1[6]	StdCPU	High Z Analog	DisableInt
Port_1_7	P1[7]	StdCPU	High Z Analog	DisableInt
Port_2_0	P2[0]	StdCPU	High Z Analog	DisableInt

Trong phần này ta chú ý các mục:

Drive:

- + High Z: Trở kháng cao, thường dùng khi làm đầu vào số.
- + High Z Analog: Thường dùng khi làm đầu vào tương tự.
- + Pull Down: Có điện trở kéo xuống.
- + Pull Up: Có điện trở kéo lên. Sử dụng khi nối với nút bấm.
- + Strong: Khi sử dụng Pin làm đầu ra.

.....

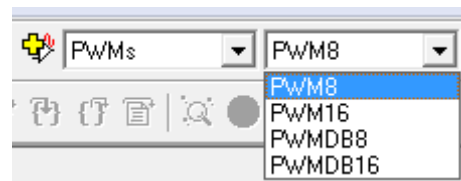
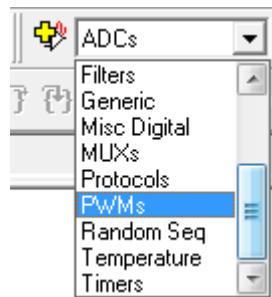
Interrupt: Sử dụng khi dùng ngắt ngoài.

- + DisableInt: Không ngắt.
- + FallingEdge: Ngắt khi có sườn xuống.
- + RisingEdge: Ngắt khi có sườn xuống.
- + ChangeFromRead: Khi có sự thay đổi mức.

- Nối các modul với các Pin.

Ví dụ như ta muốn đưa Pwm ra chân P0.0 thì ta sẽ phải làm như sau:

Trước tiên ta lấy ra bộ PWM 8 bit như sau

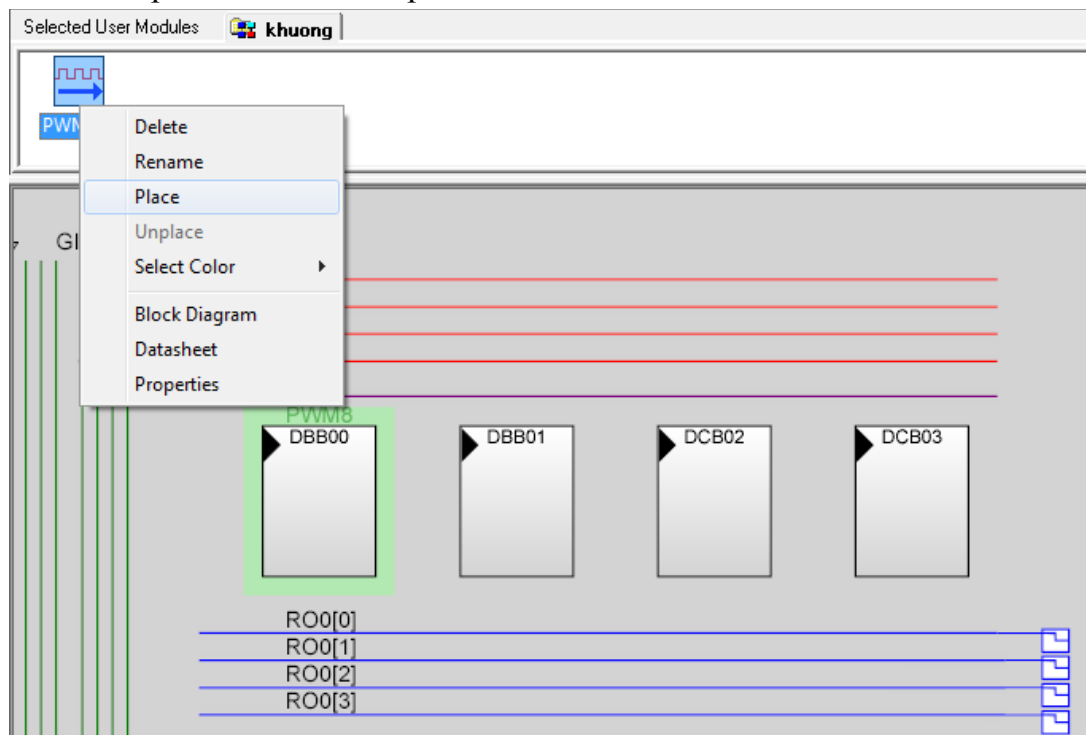


Chọn bộ Pwms

Chọn Pwm8

Sau đó Click vào biểu tượng dấu cộng bên cạnh để chuyển chúng xuống danh mục các Modul đã chọn.

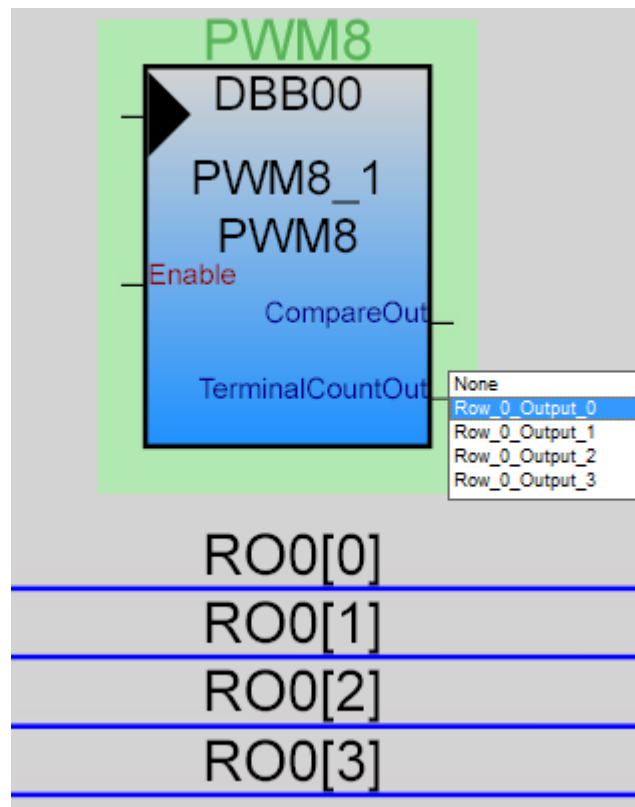
Tiếp theo chọn chuột phải và chọn Place.



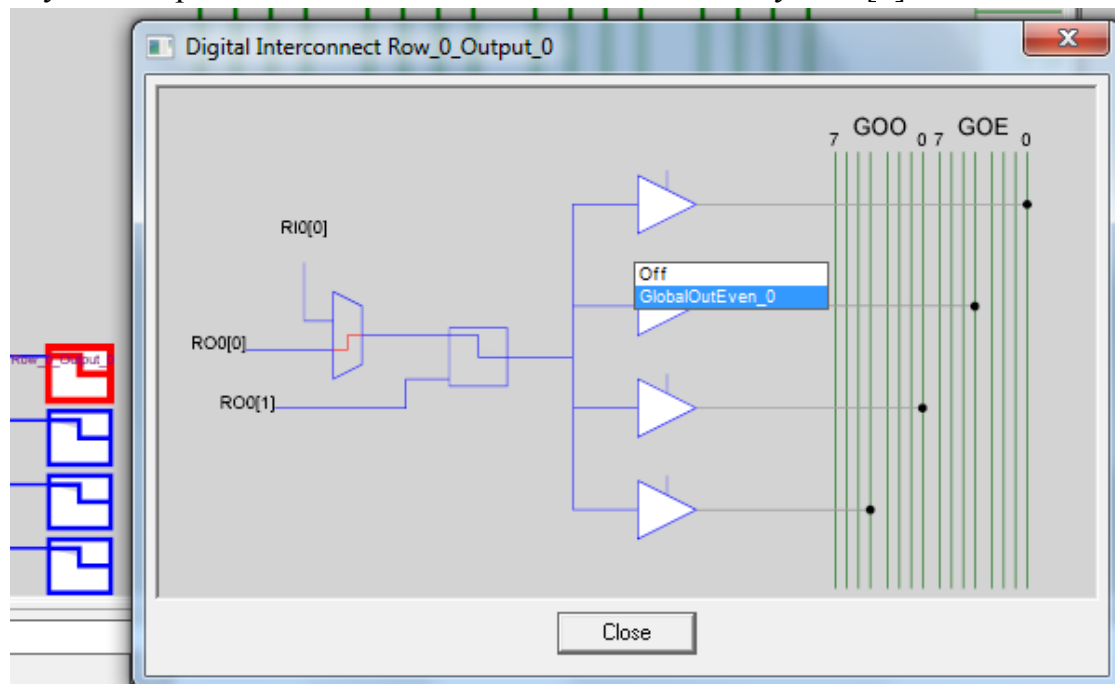
Khi đó bộ Pwm sẽ được đặt xuống vùng dành cho các khối số.

Nếu là khối tương tự như bộ adc thì sẽ được chuyển xuống vùng dành cho khối tương tự bên dưới vùng của các khối số.

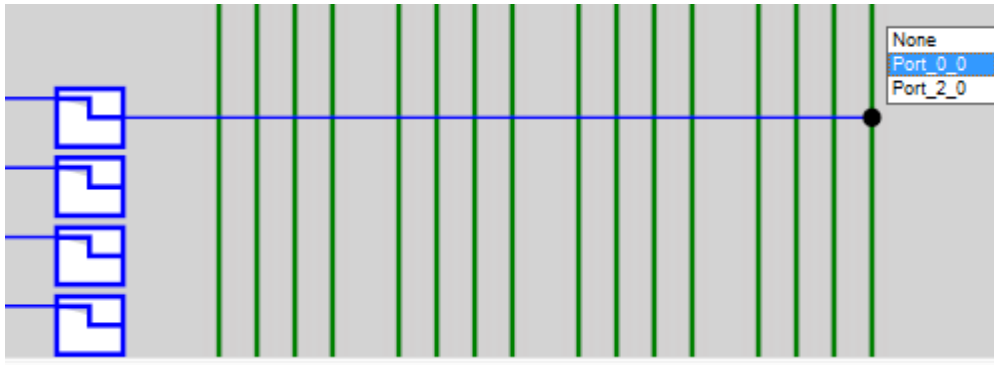
Click chuột trái vào chân
CompareOut và chọn
Row_0_Out_0 để nối với Dây
RO0[0]



Click chuột trái vào ô vuông phía trái của RO0[0] và chọn hình tam giác mà có
dây nối với pin0 sau đó chọn GlobalOutEven để nối dây RO0[0] với Pin0.

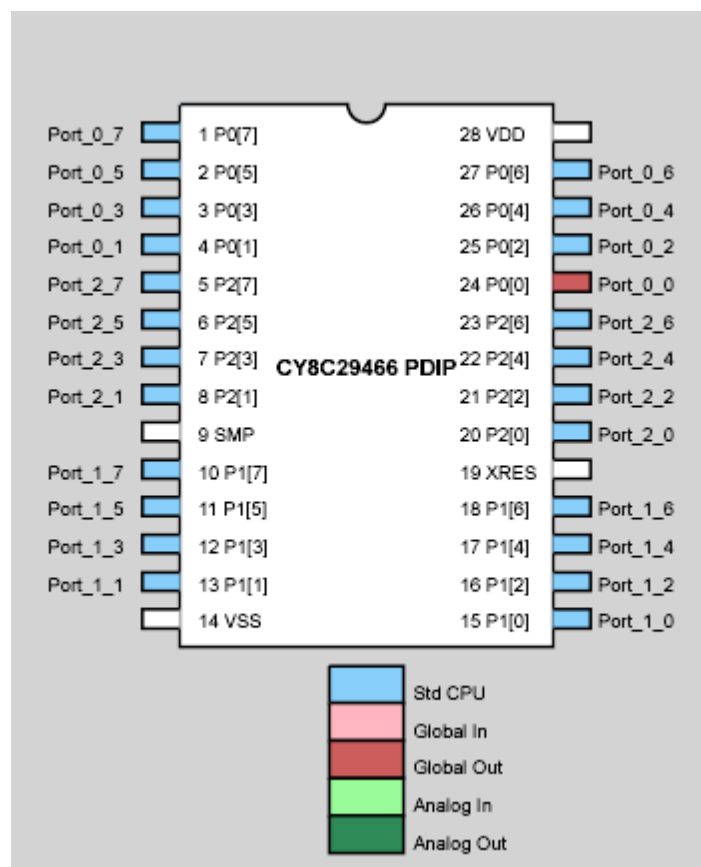


Cuối cùng chọn cột vừa được nối và chọn Port_0_0 để nối với Pin0.0



Hoàn toàn tương tự ta có thể nối với một Pin bất kỳ mà ta mong muốn. Kết nối với Pin làm đầu vào cũng làm tương tự như Pin làm đầu ra.

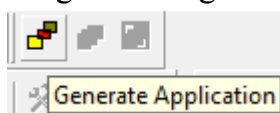
- Hình dạng chip
- Mỗi dạng cấu hình cho 1 loại tín hiệu thì sẽ được ký hiệu bởi 1 màu riêng biệt.
- Str CPU là các Pin chưa sử dụng.
- Global In: Khi pin làm đầu vào số.
- Global Out: Khi pin làm đầu ra số.
- Analog In: Pin làm đầu vào tương tự.
- Analog Out Khi làm đầu ra tương tự.
- Các chân màu trắng là chân mặc định của vi điều khiển.



Trong bài này Pin0.0 được nối với Pwm vì vậy nó là đầu ra số và có màu đỏ của Global Out.

Các thiết lập khác cho các bộ tương ứng sẽ được trình bày ở bài ví dụ cho modul đây.

Sau khi đã thiết lập xong các thông số cần thiết ta sẽ click vào biểu tượng



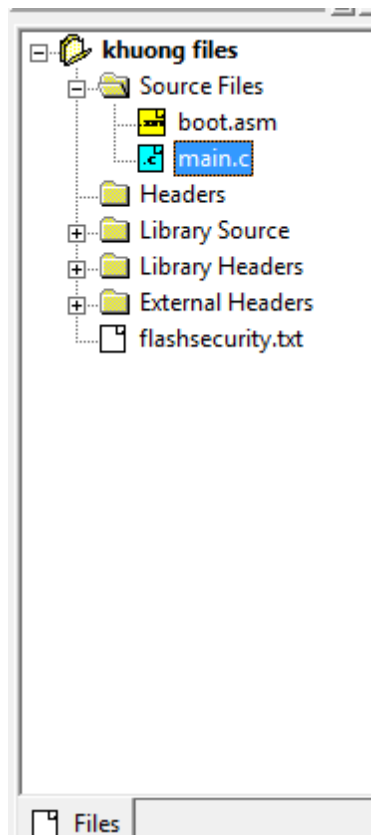
để chương trình định cấu hình cho chip.

c. Soạn thảo chương trình.

Để chuyển sang viết chương trình ta click vào biểu tượng Application Editor



Vào file – Source Files – main.c để viết chương trình.



Sau khi viết xong chương trình nhấn F7 để biên dịch. Hoặc vào Build – Build All để biên dịch.

Nếu không có lỗi gì thì 1 file.Hex được tạo ra để nạp vào chip.

```
#include <m8c.h>           // part specific constants and macros
#include "PSoC_API.h"      // PSoC API definitions for all User Modules

void delay(int t)
{
    int i,j;
    for(i =0; i<t; i++)
        for(j = 0; j<100; j++);
}

void main()
{
    PRT0DR = 0x55;
    while(1)
    {
        delay(300);
        PRT0DR = ~PRT0DR;
    }
}
```


Starting MAKE...
creating project.mk
lib/psocconfig.asm
lib/psocconfigtbl.asm
./boot.asm
./main.c

Linking..
LMM info: area 'InterruptRAM' uses 0 bytes in SRAM page 0
LMM info: area 'virtual_registers' uses 3 bytes in SRAM page 0
ROM 1% full. 514 bytes used (does not include absolute areas).
RAM 0% full. 3 bytes used (does not include stack usage).
idata dump at output/led_don.idata

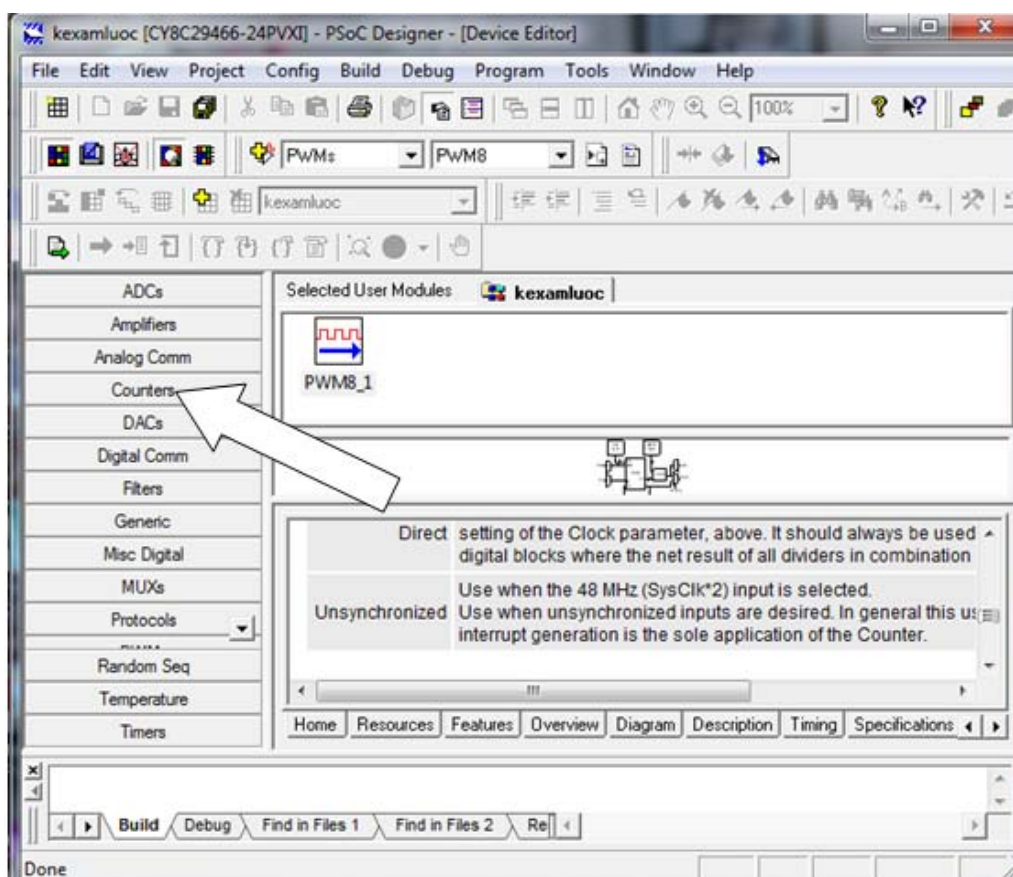
led_don - 0 error(s) 0 warning(s) 06:44:34

Build Debug Find in Files 1 Find in Files 2 Results

Nếu có lỗi thì chỉnh sửa rồi biên dịch lại.

Chú ý: Khi đang ở phần soạn thảo chương trình mà ta muốn quay lại phần thiết lập cấu hình thì chọn Device Editor .

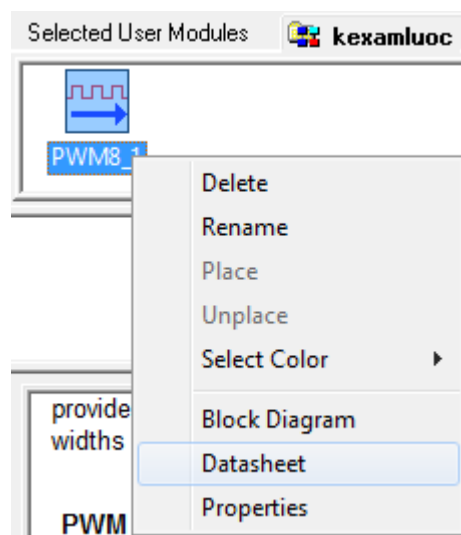
d. Tra cứu datasheet của các modul.



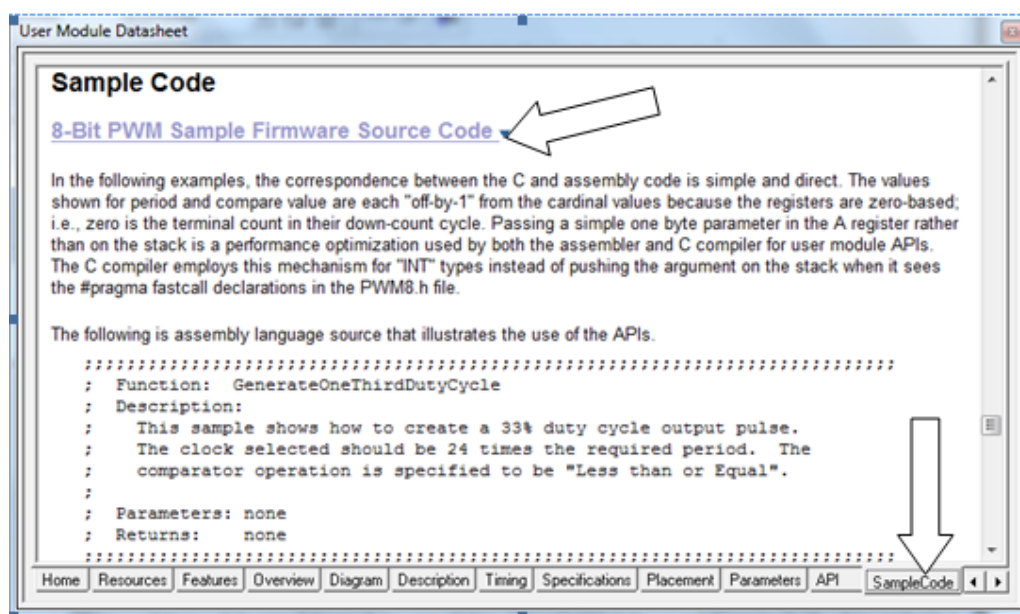
Sau khi tạo xong 1 project mới ta có giao diện như trên.

Muốn xem datasheet về modul nào thì ta có thể chọn modul đấy (như hình mũi tên). Khi đấy thông tin về modul tương ứng sẽ được hiện lên trên màn hình.

Hoặc ta có thể chọn modul rồi chọn chuột phải và chọn datasheet để xem datasheet của modul.



Thông tin về modul sẽ hiện ra như sau:



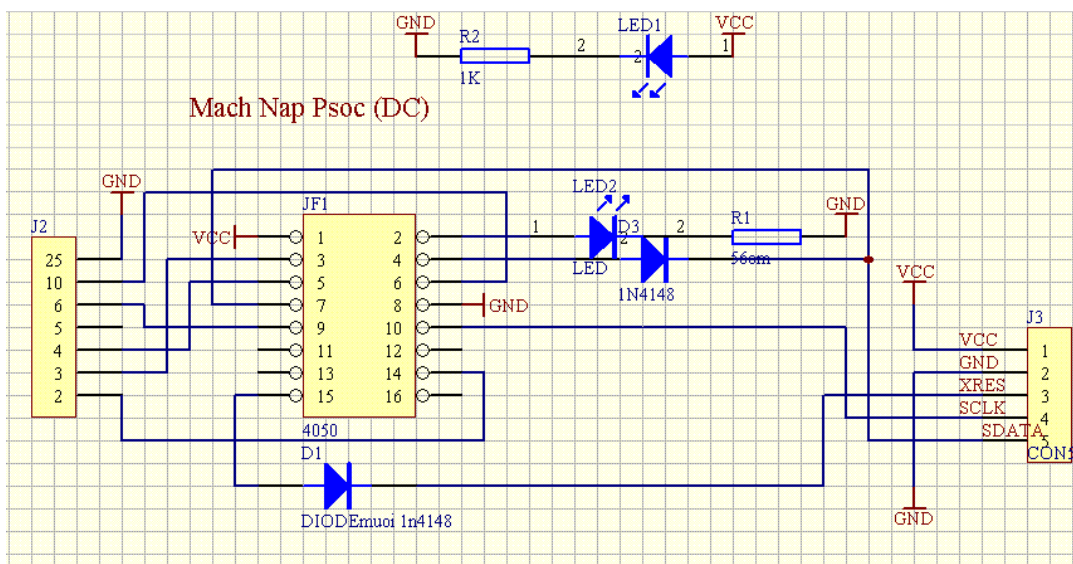
Để xem code mẫu ta chọn SampleCode.

Ngoài ra ta cũng có thể xem thêm một số thông tin khác từ phần mềm Psoc Designer khi nhấn F1.

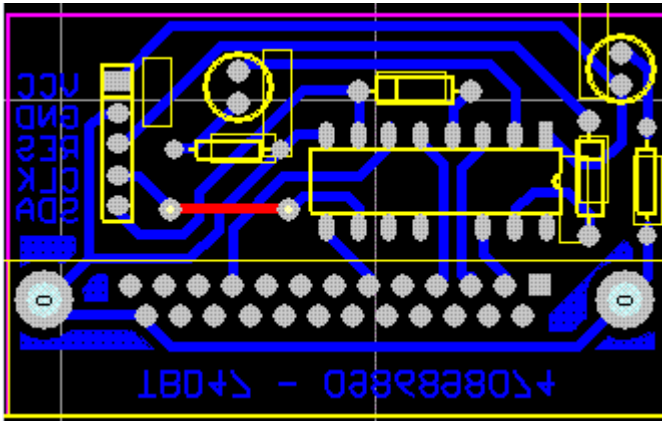
III. Mạch nạp và phần mềm nạp chương trình.

1. Mạch nạp.

- Mạch nguyên lý.



- Mạch in.



- Mạch nạp sử dụng nguồn DC chung với mạch chip chủ. Điện áp sử dụng từ 7- 24 VDC. (khi sử dụng nếu không có rắc cắm nguồn cần phân biệt rõ VDD và GND khi đấu tắt. Để phân biệt VDD và GND bạn có thể dựa vào chân tụ)
- Mạch nạp giao tiếp với máy tính bằng cổng LPT. (chú ý loại cáp cần để sử dụng là loại cáp 2 đầu đều là COM đực)
- Mạch giao tiếp với PSOC bằng cáp dây 5 sợi.

+ Sợi dây 1 là VCC (gần đèn LED nhất) nối vào VCC của chip PSOC(chân 28 loại 28 chân)

+ Sợi dây 2 là GND nối vào VSS của Psoc (chan 14 loại 28 chân)

+ Sợi dây 3 là XRES nối vào chân XRES của PSOC (chân 19 loại 28 chân)

+ Sợi dây 4 là SCLK là chân tín hiệu clock nối vào chân SCLK(chân 13 loại 28 chân)

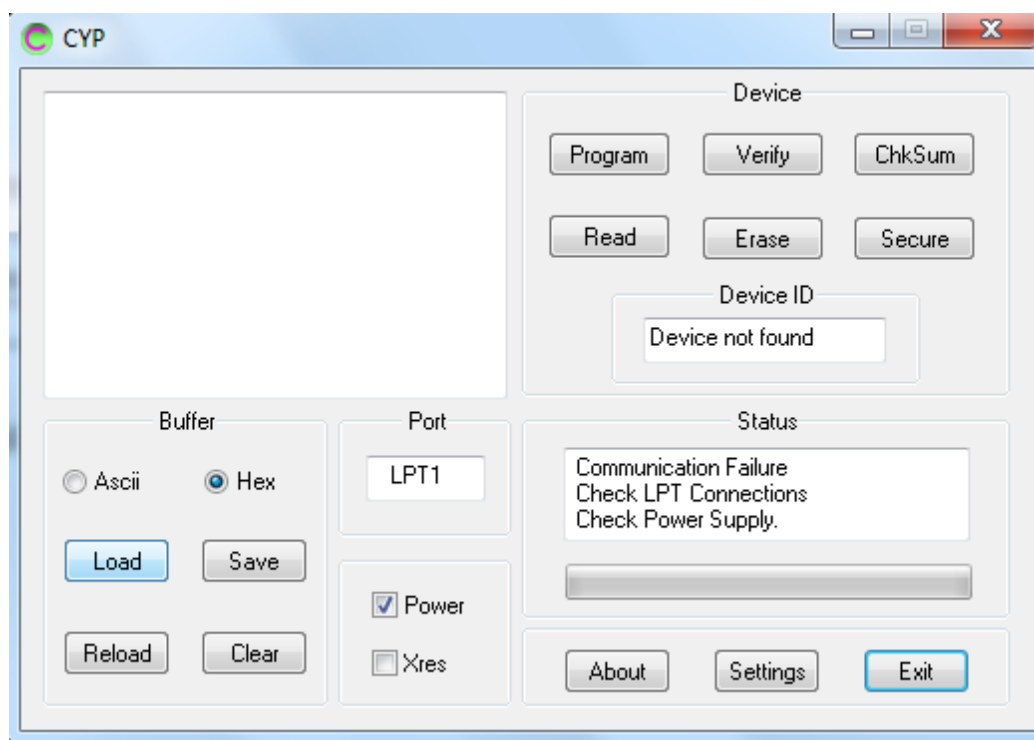
+ Sợi dây 5 là SDA là chân data nối vào chân SDA(chân 15 loại 28 chân)

- Các bạn khi sử dụng chú ý đấu đúng thứ tự dây.

2. Hướng dẫn sử dụng mạch nạp PSOC và chương trình nạp.

Để nạp chương trình ta sử dụng phần mềm CyP.

Sau khi chạy phần mềm có giao diện như sau.



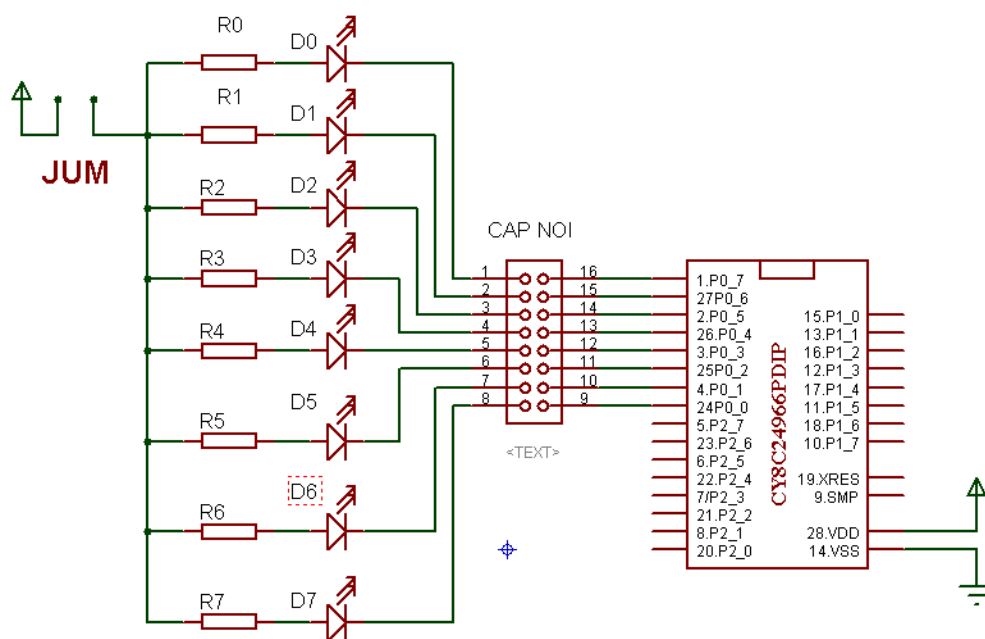
- Cắm chân nạp của mạch nạp vào chip đúng chiều sao cho Led trên modul CPU và trên mạch nạp đều sáng đẹp. Nếu đèn sáng mờ là sai chiều.
- Kiểm tra xem đã nhận chip chưa bằng cách click vào CheckSum. Nếu đã kết nối thì tên chip sẽ được hiển thị trong mục Device ID.
- Chọn file cần nạp bằng cách click vào Load, sau đó dẫn đến file cần nạp.
- Nạp chương trình: Click Program.
- Sau khi nạp báo thành công Click đúp vào Xres để chạy chương trình.

IV. Các bài thực hành.

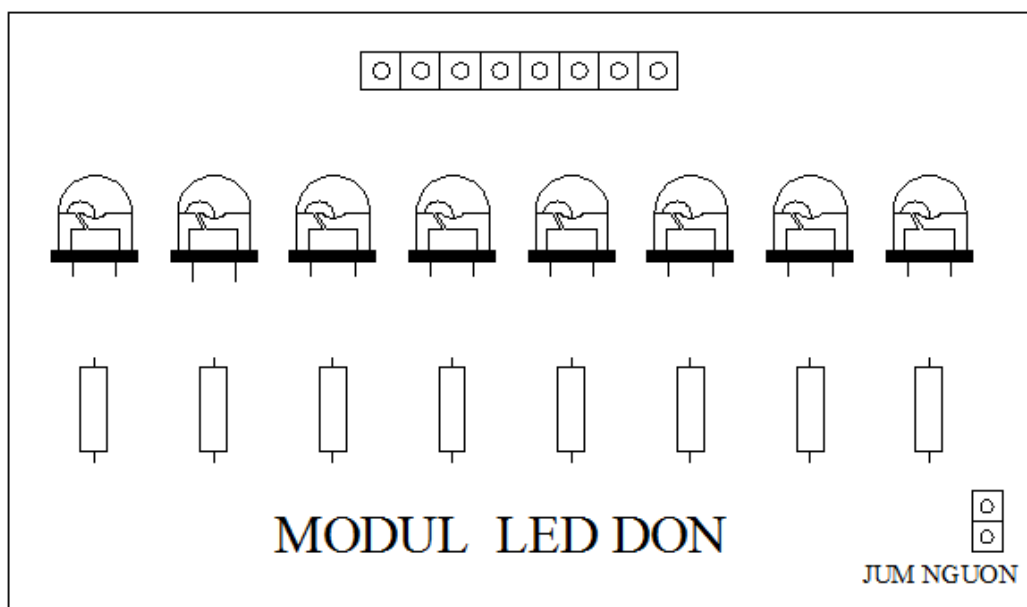
Bài 1: Led đơn.

Bài này sẽ sử dụng Port 0 của vi điều khiển để điều khiển 8 led đơn. Ta sẽ xuất tín hiệu ra các chân của vi điều khiển để điều khiển các Led. Vì vậy các chân này cần được cấu hình là chân đầu ra Strong.

1. Mạch nguyên lý.



Mạch trên Kit.



2. Kết nối cáp.

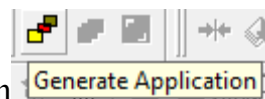
Nối Jum cấp nguồn cho Modul Led, và nối cáp giữa Port0 của vi điều khiển với các led.

3. Viết chương trình.

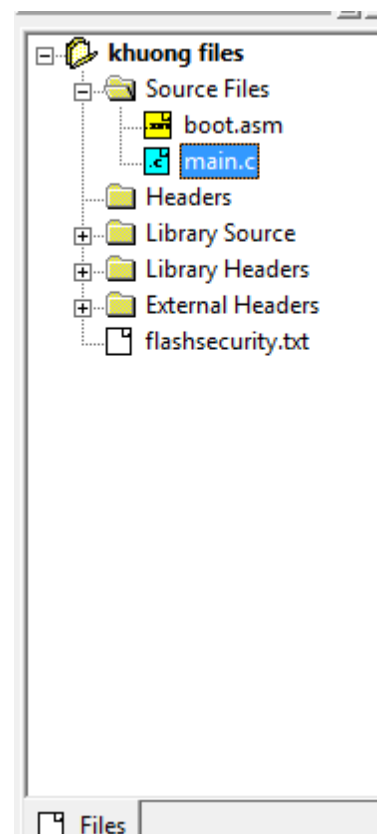
Sau khi tạo project mới click vào Interconnct View để chuyển sang thiết lập cấu hình cho các chân của vi điều khiển.

Trong mục Drive ta chọn các chân của Port 0 là Strong (Sử dụng port 0 là đầu ra).

Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	High Z Analog	DisableInt
Port_0_1	P0[1]	StdCPU	High Z	DisableInt
Port_0_2	P0[2]	StdCPU	High Z Analog	DisableInt
Port_0_3	P0[3]	StdCPU	Open Drain High	DisableInt
Port_0_4	P0[4]	StdCPU	Open Drain Low	DisableInt
Port_0_5	P0[5]	StdCPU	Pull Down	DisableInt
Port_0_6	P0[6]	StdCPU	Pull Up	DisableInt
Port_0_7	P0[7]	StdCPU	Strong	DisableInt
Port_1_0	P1[0]	StdCPU	Strong Slow	DisableInt
Port_1_1	P1[1]	StdCPU	High Z Analog	DisableInt
Port_1_2	P1[2]	StdCPU	High Z Analog	DisableInt
Port_1_3	P1[3]	StdCPU	High Z Analog	DisableInt



Tiếp theo chọn Generate Application để xác lập cấu hình.
Để chuyển sang viết chương trình ta click vào biểu tượng Application Editor



Vào file – Source Files – main.c để viết chương trình.

Soạn thảo chương trình.

// Nháy led đơn

// 14.6.2011

// PRT0DR được cấu hình là Strong, Nối với led đơn.


```
#include <m8c.h>      // part specific constants and macros
#include "PSoCAPI.h"  // PSoC API definitions for all User Modules

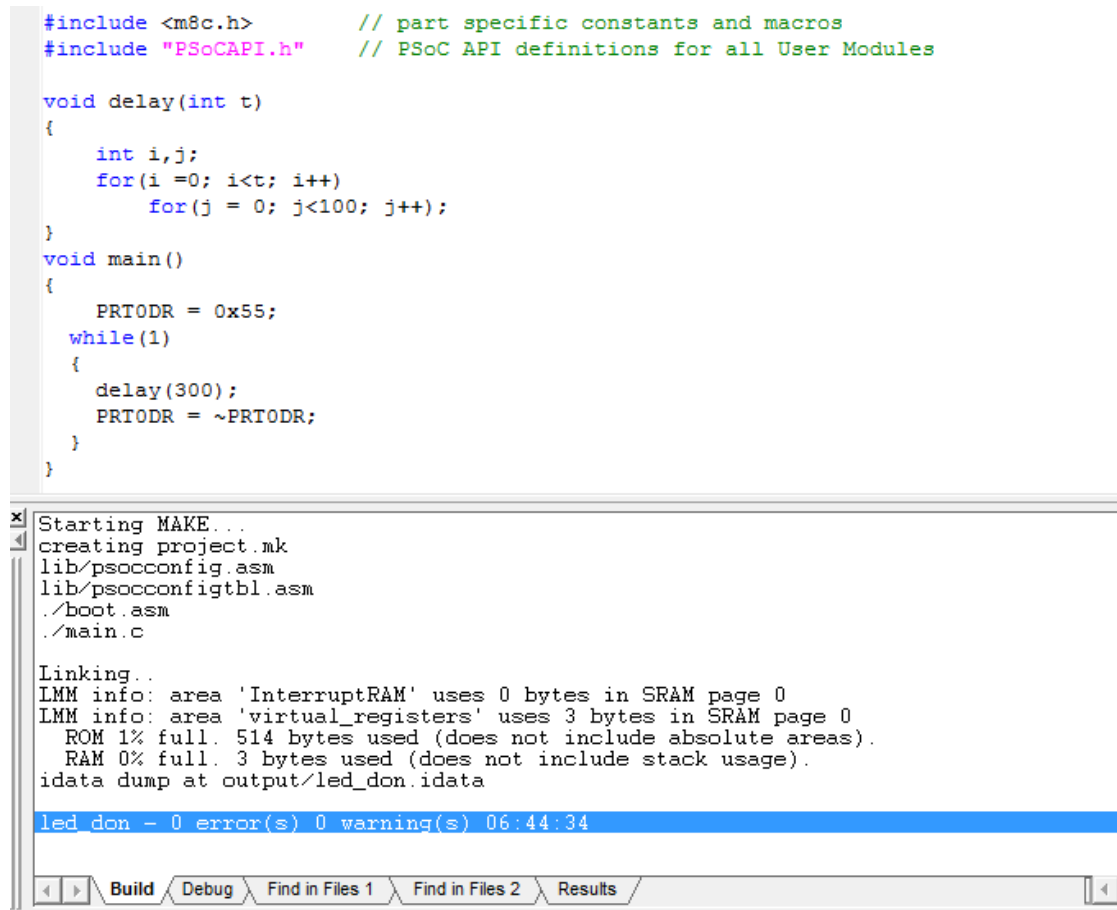
void delay(int t)
{
    int i,j;
    for(i=0; i<t; i++)
        for(j = 0; j<100; j++);
}

void main()
{
    PRT0DR = 0x55;
    while(1)
    {
        delay(300);
        PRT0DR = ~PRT0DR;
    }
}
```

Sau khi viết xong chương trình nhấn F7 để biên dịch.

Nếu không có lỗi gì thì 1 file.Hex được tạo ra để nạp vào chip.

Nếu có lỗi thì chỉnh sửa rồi biên dịch lại.



```
#include <m8c.h>      // part specific constants and macros
#include "PSoCAPI.h"  // PSoC API definitions for all User Modules

void delay(int t)
{
    int i,j;
    for(i=0; i<t; i++)
        for(j = 0; j<100; j++);
}

void main()
{
    PRT0DR = 0x55;
    while(1)
    {
        delay(300);
        PRT0DR = ~PRT0DR;
    }
}
```

```
Starting MAKE...
creating project.mk
lib/psocconfig.asm
lib/psocconfigtbl.asm
./boot.asm
./main.c

Linking..
LMM info: area 'InterruptRAM' uses 0 bytes in SRAM page 0
LMM info: area 'virtual_registers' uses 3 bytes in SRAM page 0
ROM 1% full. 514 bytes used (does not include absolute areas).
RAM 0% full. 3 bytes used (does not include stack usage).
idata dump at output/led_don.idata

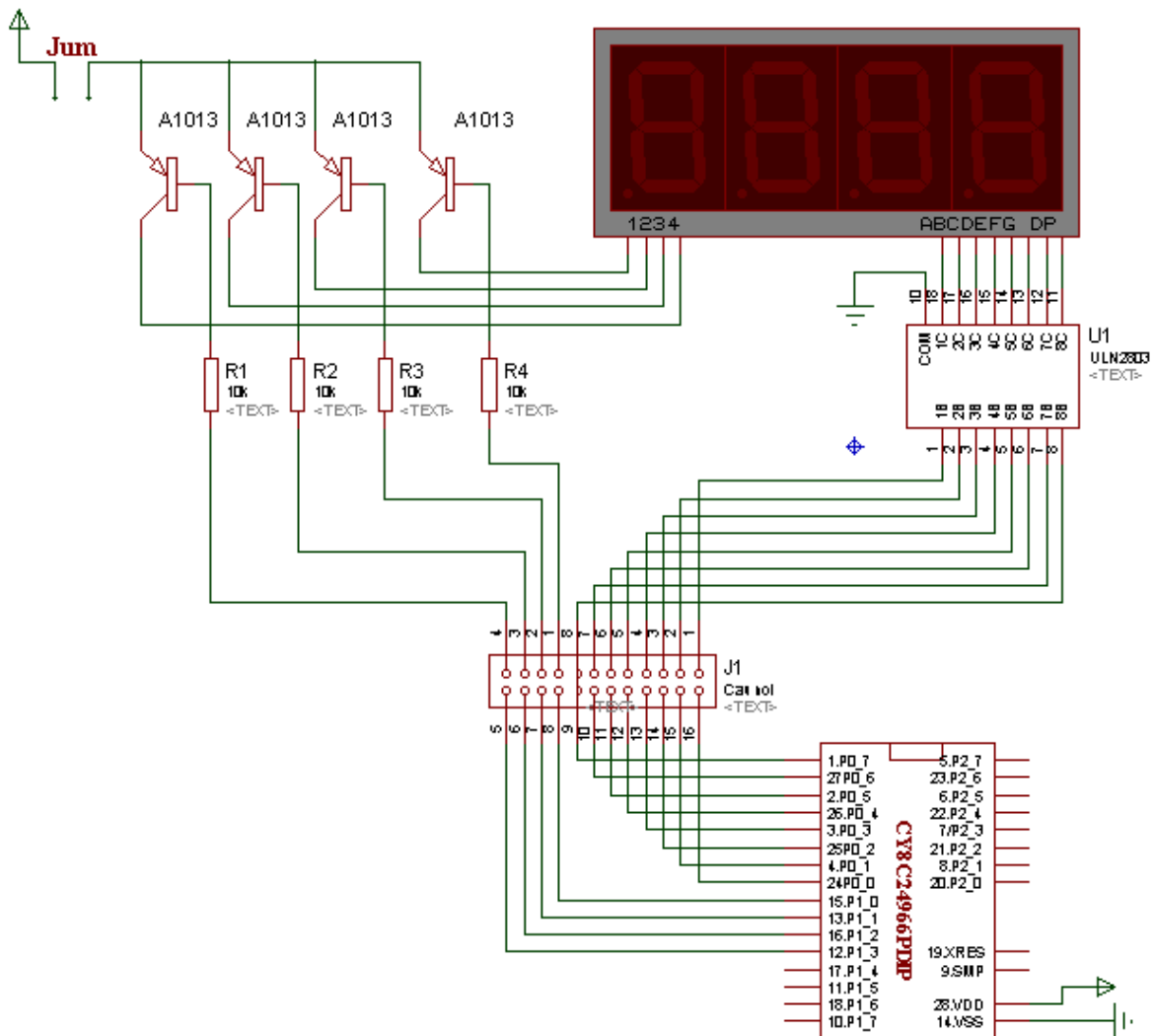
led_don - 0 error(s) 0 warning(s) 06:44:34
```

Build Debug Find in Files 1 Find in Files 2 Results

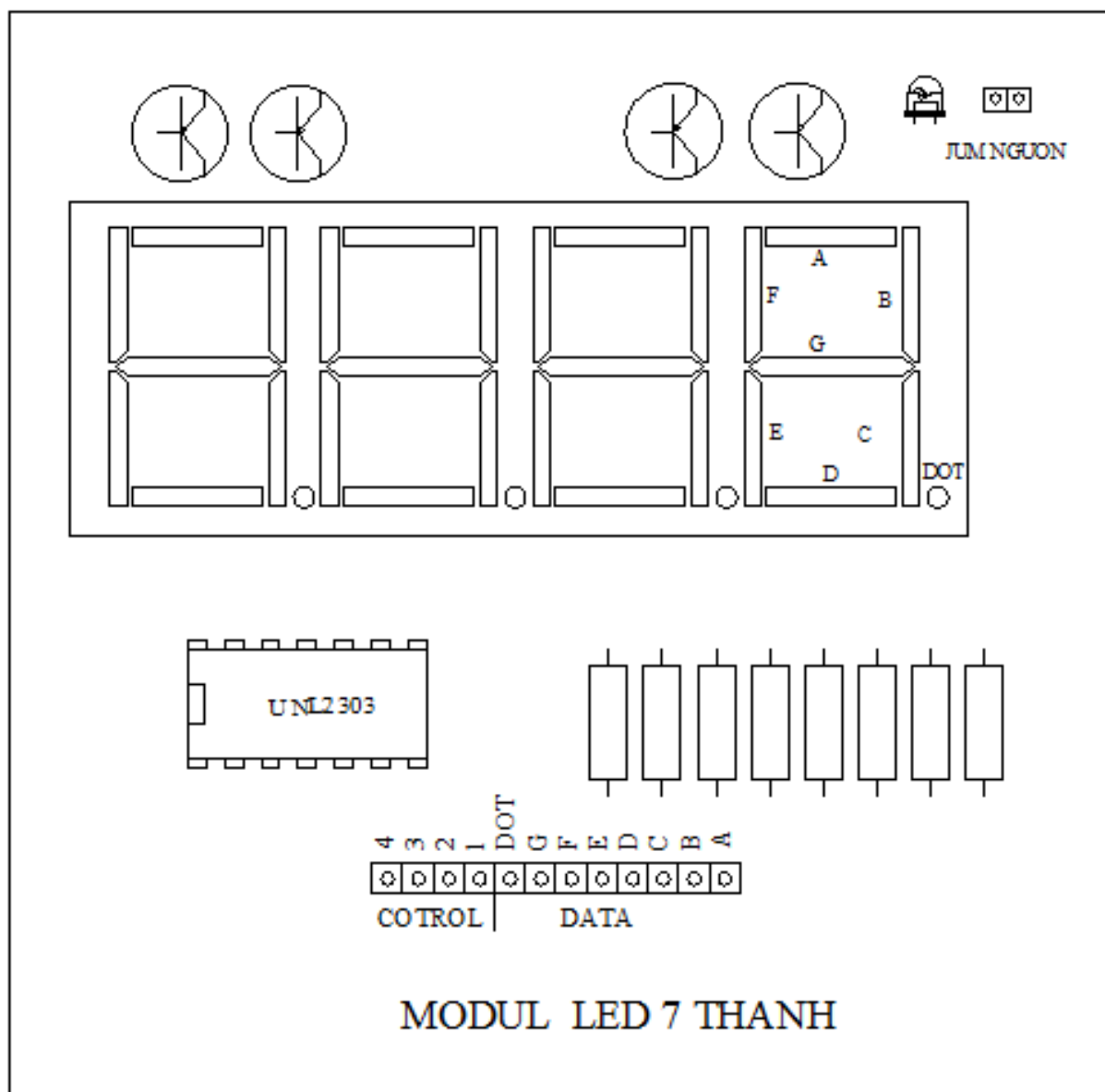
Bài 2. Led 7 thanh.

Bài này sẽ thực hiện hiển thị từ 0 đến 9999 trên led 7 thanh 4 trong 1 anode chung.

1. Sơ đồ mạch nguyên lý.



Mạch trên kit.



2. Nối cáp.

Nối Jum cấp nguồn cho modul.

Nối cáp giữa vi điều khiển với Led 7 thanh theo đúng thứ tự.

Chân P0.0 nối với a, P0.1 với b..... P0.7 với dot

Chân P1.0 nối với chân control của led thứ nhất P1.3 nối với chân control của led thứ nhất 4.

3. Viết chương trình.

Tương tự như bài led đơn, cấu hình tất cả các chân là Strong.

// quet led 7 thanh dem tu 0 den 9999

// 14.6.2011

// chan data: P0.0 - a, P0.1 - b, P0.2 - c, P0.3 - d, P0.4 - e, P0.5 - f, P0.7 - g,

// P0.7 - dot.

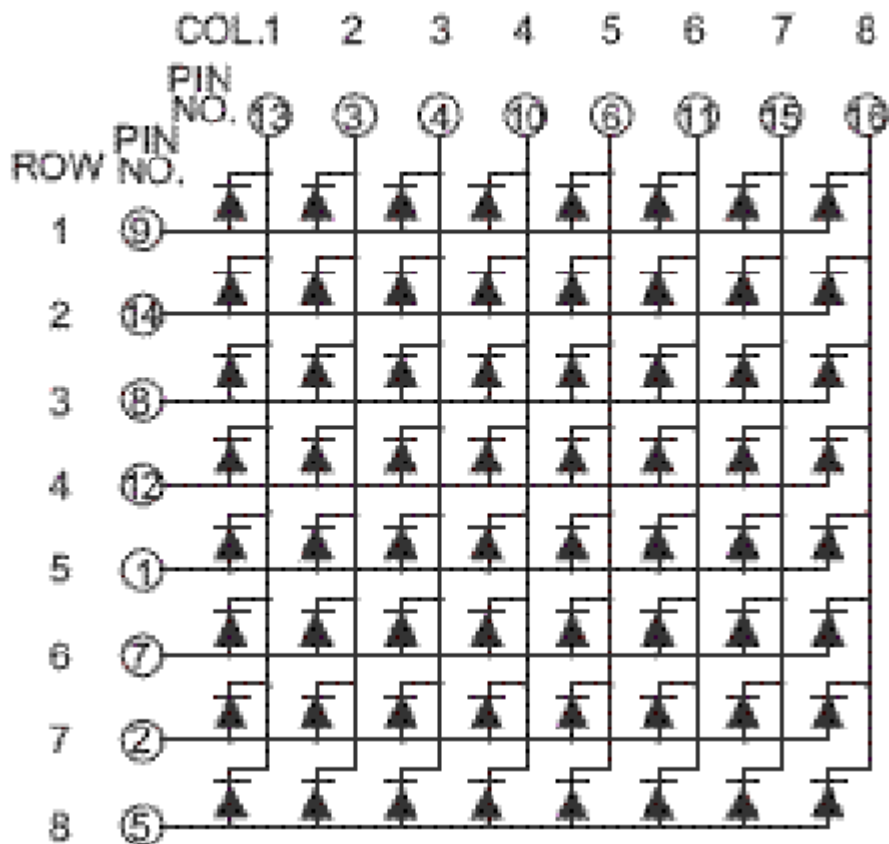
```
// chan control: P1.0 - 1, P1.1 - a2, P1.3 - 3, P1.4 - 4.  
// chan data dieu khien muc 1, chan control dieu khien muc 0.  
// cau hinh PRT0DR va PRT1DR la Strong.
```

```
#include <m8c.h>  
#include "PSoCAPI.h"  
Unsigned char ma[10] =  
{0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6}; // tinh muc 1.  
unsigned int lap,so=0;  
void delay(int t)  
{  
    int i,j;  
    for(i =0; i<t; i++)  
        for(j = 0; j<100; j++);  
}  
void quetled(unsigned int k)  
{  
    // hien thi len led 1  
    PRT1DR = 0xfe;  
    PRT0DR = ma[((k%1000)%100)%10];  
    PRT1DR = 0xff;  
    delay(1);  
  
    // hien thi len led 2  
    PRT1DR = 0xfd;  
    PRT0DR = ma[((k%1000)%100)/10];  
    PRT1DR = 0xff;  
    delay(1);  
  
    // hien thi len led 3  
    PRT1DR = 0xfb;  
    PRT0DR = ma[(k%1000)/100];  
    PRT1DR = 0xff;  
    delay(1);  
  
    // hien thi len led 4  
    PRT1DR = 0xf7;  
    PRT0DR = ma[k/1000];  
    PRT1DR = 0xff;  
    delay(1);  
}
```

```
void main()
{
    PRT1DR = 0xff;
    while(1)
    {
        if(lap++>3000)
        {
            lap = 0;
            so++;
        }
        if(so>9999)
            so = 0;
        quetled(so);
    }
}
```

Bài 3. Ma trận led.

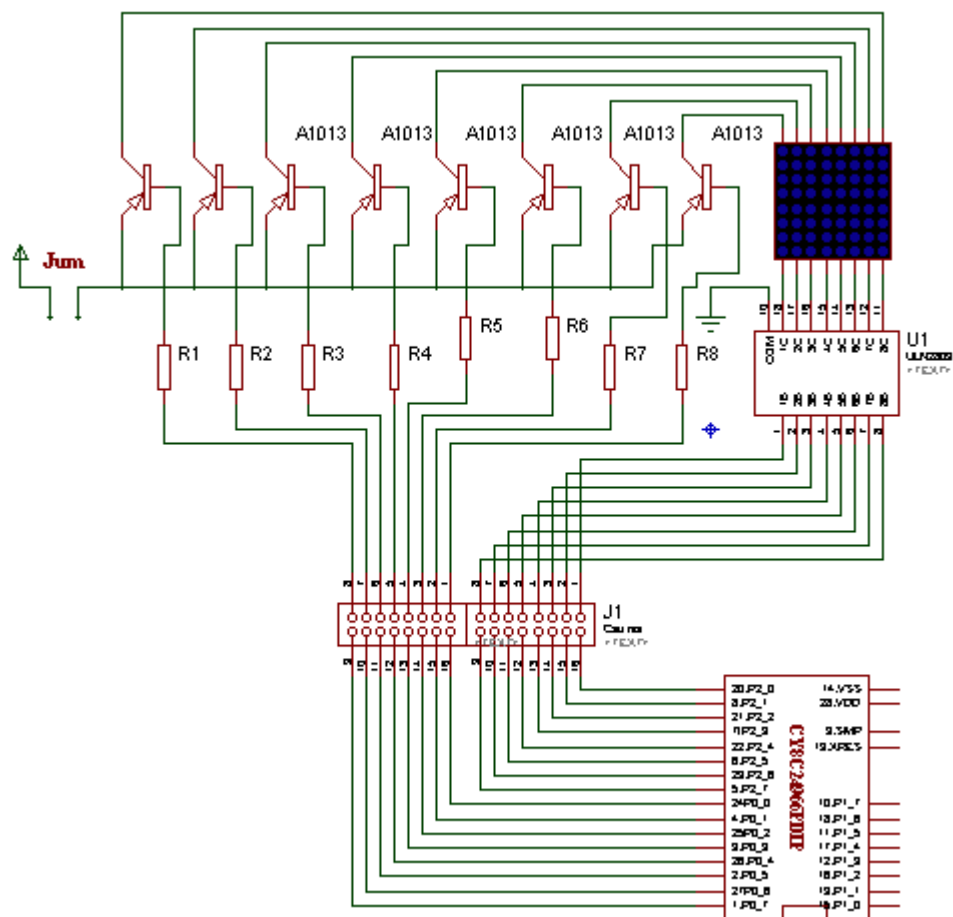
1. Cấu tạo ma trận led.



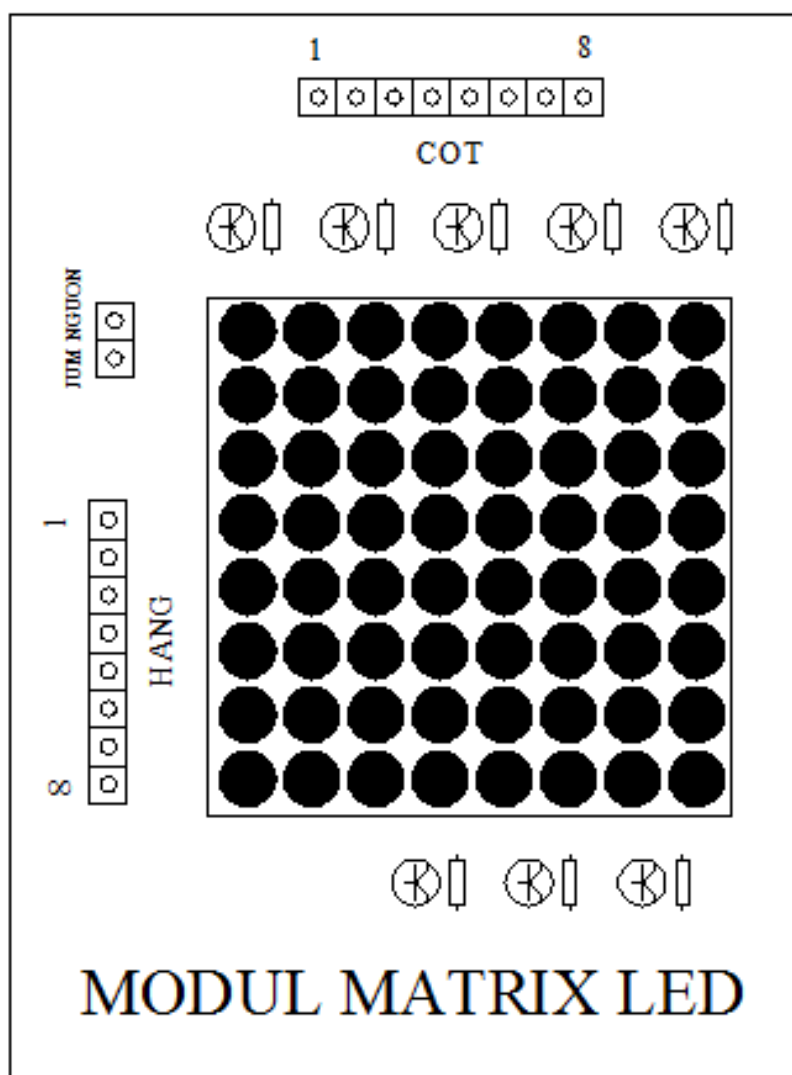
Ma trận Led 8x8. Các Led được nối với nhau thành 8 hàng và 8 cột.

Muốn cho 1 led sáng thì ta phải đưa hàng của nó lên 1 và cột tương ứng xuống 0.

2. Sơ đồ nguyên lý.



3. Mạch trên kit.



4. Nối cáp.

Nối Port2 của VDK với các hàng của ma trận led.

Nối Port0 của VDK với các cột của ma trận led.

5. Viết chương trình.

Cấu hình tất cả các chân Port 0, 2 là Strong.

// 14.6.2011

// Quét theo hàng

// Hiện thị lần lượt từ 0 đến 9

// Data tính 0 nối vào port 0 VDK và cột của matrix

// Control tính 1 nối vào port2 của VDK và hàng của matrix

// Tất cả các chân của Port0, Port2 đều Strong.

```
#include <m8c.h>
```

```
#include "PSoC_API.h"
```

```
unsigned char control[8] = { 0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x08};
```

```
unsigned char data [80] = {
```

```

0xff,0x81,0x7e,0x7e,0x7e,0x7e,0x81,0xff,//0

0xff,0x7b,0x7d,0x00,0x00,0x7f,0x7f,0xff,//1

0x3d,0x5e,0x6e,0x76,0x39,0xff,0xff,0xff,//2

0xff,0xbd,0x76,0x76,0x76,0x89,0xff,0xff,//3

0xef,0xe7,0xeb,0x6d,0x00,0x6f,0xef,0xff,//4

0xff,0x78,0x7a,0x7a,0xba,0xc6,0xff,0xff,//5

0xff,0x81,0x76,0x76,0x76,0x8d,0xff,0xff,//6

0xff,0xff,0xfc,0xfe,0xfe,0xfe,0x00,0xff,//7

0xff,0x89,0x76,0x76,0x76,0x89,0xff,0xff,//8

0xff,0xb9,0x76,0x76,0x76,0x81,0xff,0xff //9
};

```

```

void delay(unsigned int t)
{
    unsigned int i,j;
    for(i=0;i<t;i++)
        for(j=0;j<100;j++);
}

void matrixled(int k)
{
    int n;
    PRT0DR = 0xFF;
    PRT2DR = 0x00;

    for(n=0;n<8;n++)
    {
        PRT0DR = data[k*8 + n];
        PRT2DR = control[n];
        delay(1);
    }
}

```

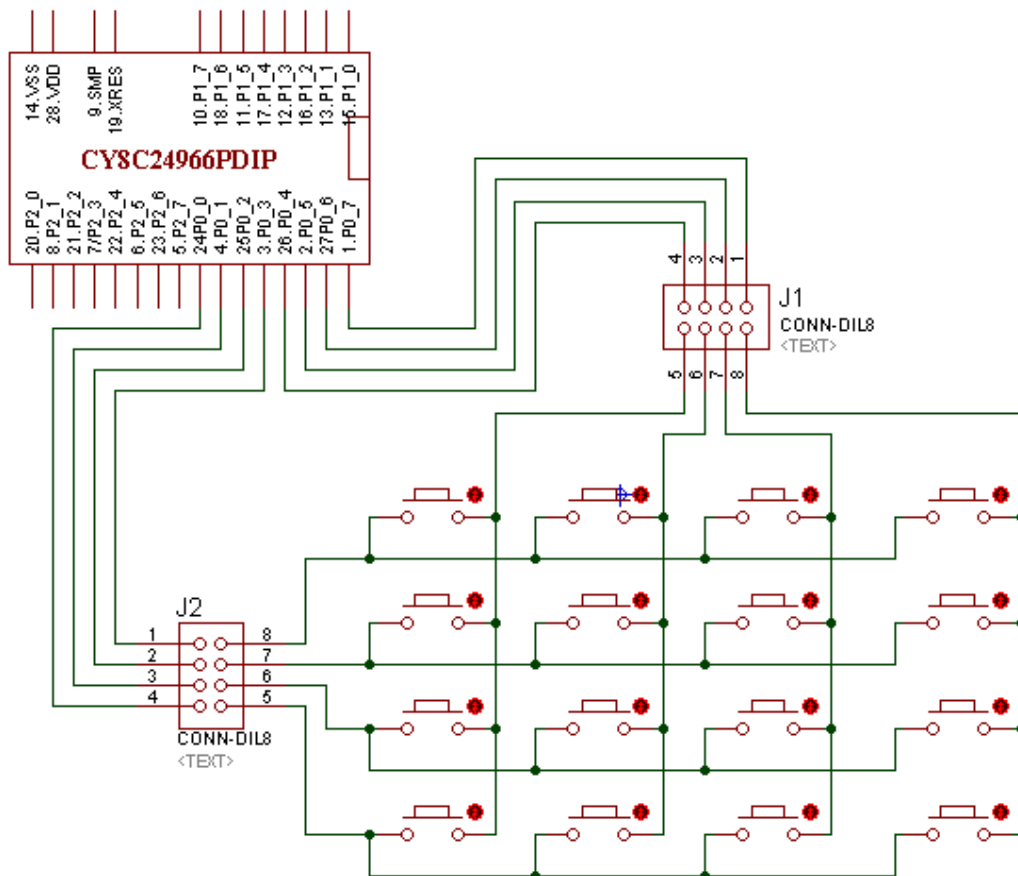


```
void main()
{
    int chay = 0,lap;
    PRT0DR = 0xFF;
    PRT2DR = 0xFF;

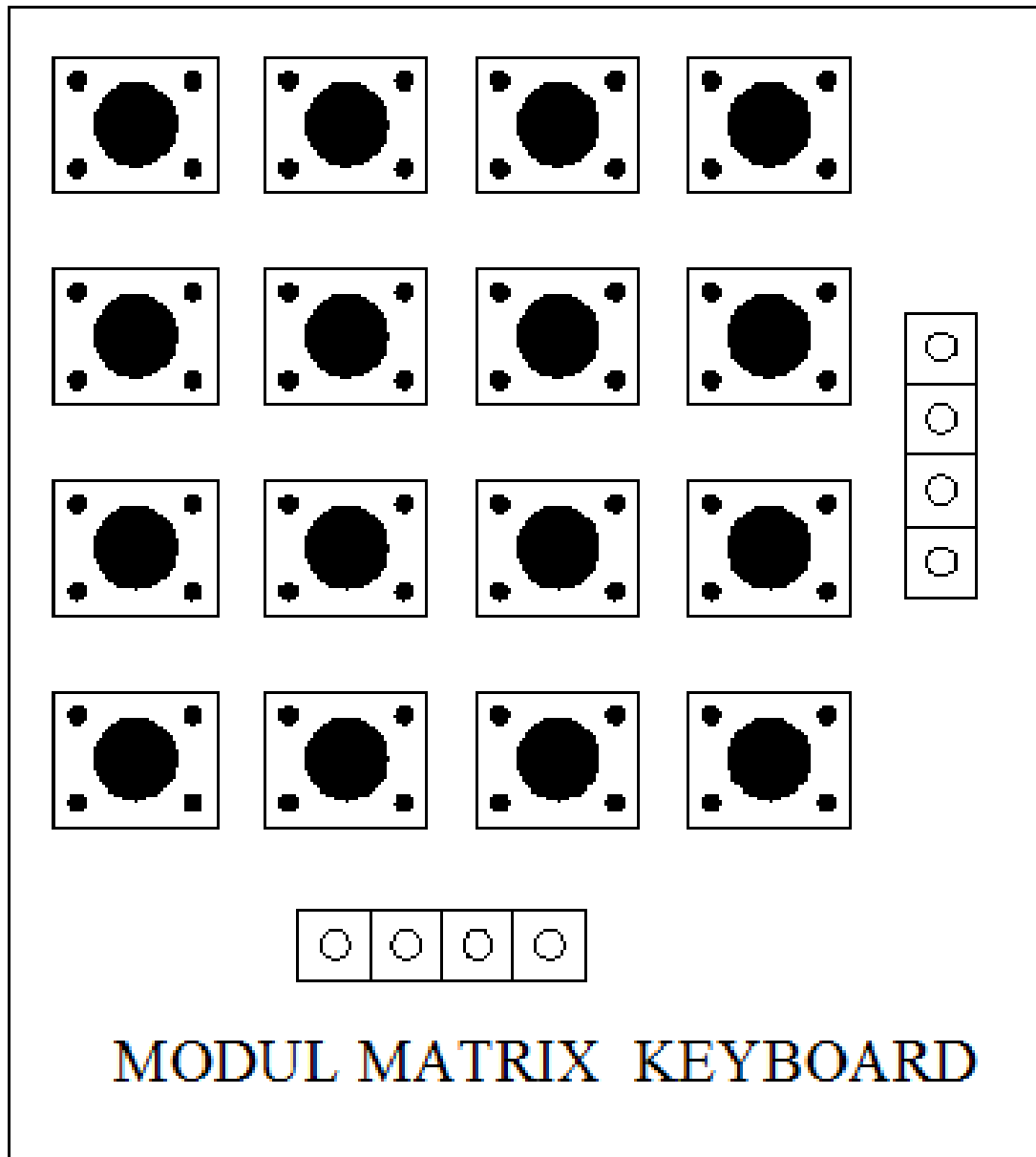
    while(1)
    {
        for(chay=0;chay<10;chay++)
        {
            for(lap=0;lap<500;lap++)
            {
                matrixled(chay);
            }
            PRT0DR = 0xFF;
            PRT2DR = 0x00;
        }
    }
}
```

Bài 4. Ma trận bàn phím

1. Sơ đồ nguyên lý.



2. Mạch trên Kit.



3. Nối cáp.

Cấu hình P2.0 đến P2.3 là Strong nối với các hàng.

P2.4 đến P2.7 là Pull Up nối với các cột.

Nối các chân Port 0 với các chân data của led 7 thanh

Nối 4 chân thấp của port 1 với chân control của led 7 thanh để hiển thị phím bấm.

Các chân của Port 1 và 0 cấu hình là Strong.

4. Chương trình.

Code chương trình.

//ma tran phim 4x4.

// port 0 data cho led 7seg. port1 control cho 7seg P1.0 va P1.1.

// port2 control matrix keyboard(0==>3 strong, 4==>7 pull up.

```
#include <m8c.h>      // part specific constants and macros
#include "PSoC_API.h" // PSoC API definitions for all User Modules
unsigned char data[10] =
{0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6}; // tinh muc 1.
unsigned int phim;
void delay(unsigned int t)
{
    unsigned int i,j;
    for(i=0;i<t;i++)
        for(j=0;j<100;j++);
}

void quetled( unsigned int k)
{
    PRT1DR = 0xfe;
    PRT0DR = data[k/10];
    //PRT1DR = 0xFF;
    delay(1);

    PRT1DR = 0xfd;
    PRT0DR = data[k%10];
    //PRT1DR = 0xFF;
    delay(1);
}

void quetphim()
{
    PRT2DR = 0xFE;
    if((PRT2DR & 0x10) == 0)    phim = 0;
    if((PRT2DR & 0x20) == 0)    phim = 4;
    if((PRT2DR & 0x40) == 0)    phim = 8;
    if((PRT2DR & 0x80) == 0)    phim = 12;
    delay(1);

    PRT2DR = 0xFD;
    if((PRT2DR & 0x10) == 0)    phim = 1;
    if((PRT2DR & 0x20) == 0)    phim = 5;
    if((PRT2DR & 0x40) == 0)    phim = 9;
    if((PRT2DR & 0x80) == 0)    phim = 13;
    delay(1);
}
```

```

PRT2DR = 0xFB;
if((PRT2DR & 0x10) == 0)    phim = 2;
if((PRT2DR & 0x20) == 0)    phim = 6;
if((PRT2DR & 0x40) == 0)    phim = 10;
if((PRT2DR & 0x80) == 0)    phim = 14;
delay(1);

PRT2DR = 0xF7;
if((PRT2DR & 0x10) == 0)    phim = 3;
if((PRT2DR & 0x20) == 0)    phim = 7;
if((PRT2DR & 0x40) == 0)    phim = 11;
if((PRT2DR & 0x80) == 0)    phim = 15;
delay(1);

}

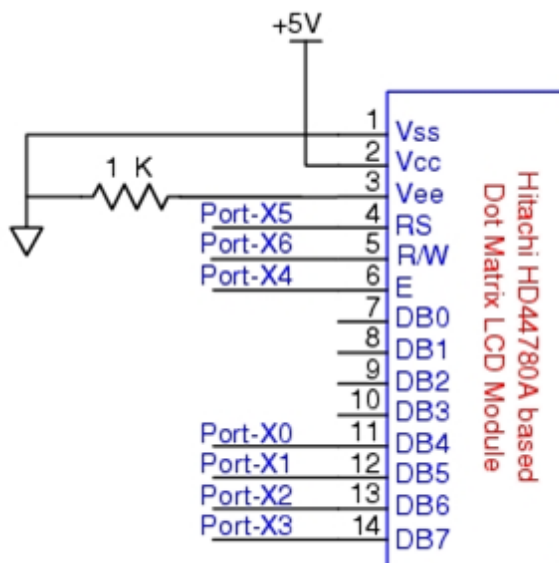
void main()

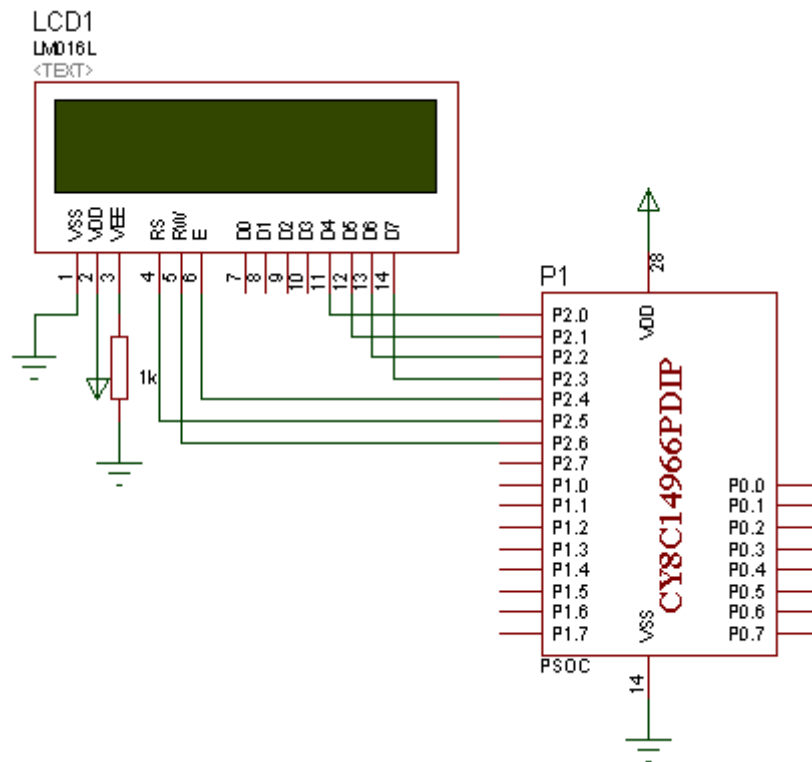
{
    while(1)
    {
        quetphim();
        quetled(phim);
    }
}

```

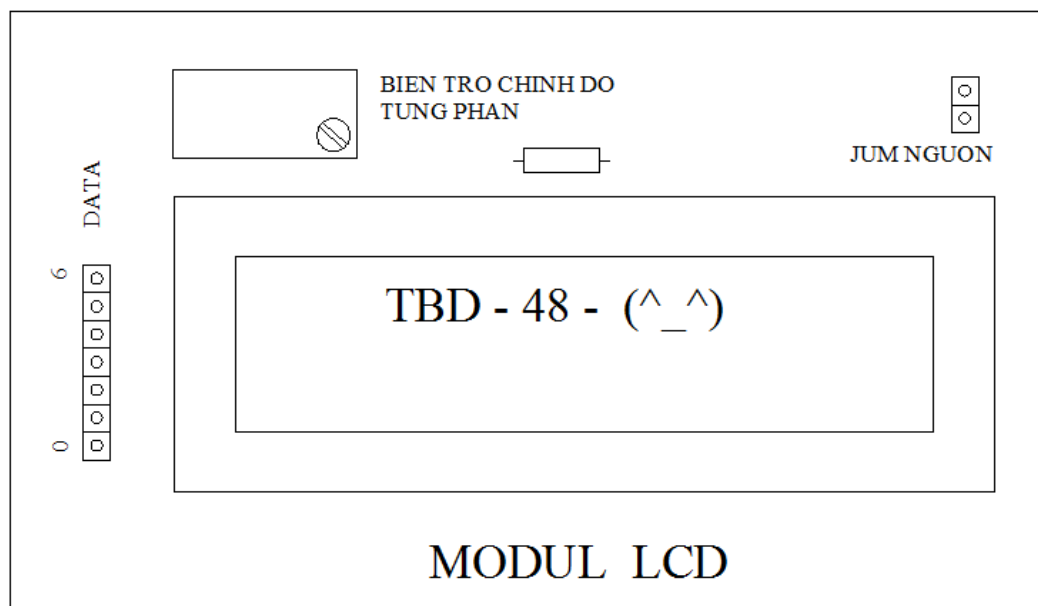
Bài 5. LCD

1. Sơ đồ nguyên lý.





2. Mạch trên kit.



3. Nối cáp.

Nối cáp giữa port 2 và LCD theo đúng thứ tự như số đánh trên kit.

4. Code chương trình.

Để chọn khối LCD ta chuyển sang thiết lập phần cứng và chọn như sau:



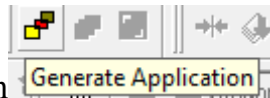
Khối LCD nằm trong mục Misc Digital.

Sau khi chọn xong ta click vào biểu tượng dấu cộng để xác lập cấu hình.

Chọn Port kết nối với LCD

LCD_1	
User Module Parameters	Value
LCDPort	Port_2
BarGraph	None
	Port_0
	Port_1
	Port_2

Sau khi chọn xong port cho LCD chọn Generate



Application để xác lập cấu hình. Rồi chuyển sang giao diện viết chương trình.

```
// LCD hien thi so, ky tu, chuoi ky tu.
// 14.06.2011
//hien thi so tu 0 den 99
// su dung PRT2DR cho LCD
#include <m8c.h>      // part specific constants and macros
#include "PSoCAPI.h"  // PSoC API definitions for all User Modules
```

```
void Hienthi(int data)
{
    LCD_1_WriteData(data/10+48);
    LCD_1_WriteData(data%10+48);
}
```

```
void delay(int t)
{
    unsigned int i,j;
    for(i=0;i<t;i++)
        for(j=0;j<100;j++);
}
```

```
void main()
{
    // Insert your main routine code here.
    int chay;
    LCD_1_Start();
    while(1)
    {
        LCD_1_Position(0,3);
        LCD_1_PrCString("Dem:");

        for(chay=0;chay<100;chay++)
        {
```

```

LCD_1_Position(0,8);
Hienthi(chay);
delay(50);
LCD_1_Position(1,0);
LCD_1_PrCString("TBD48-KXL_(^-^)");
}

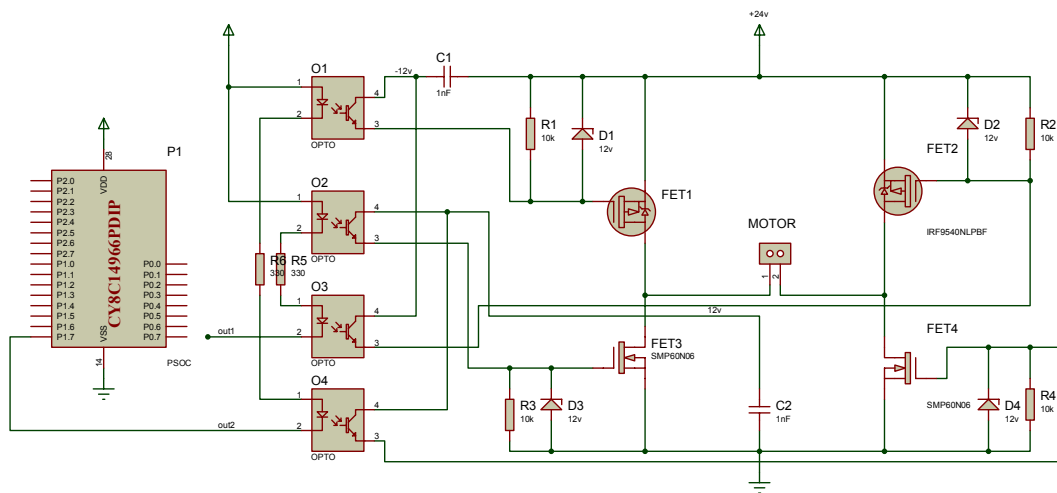
}

}

```

Bài 6. Mạch cầu H - PWM.

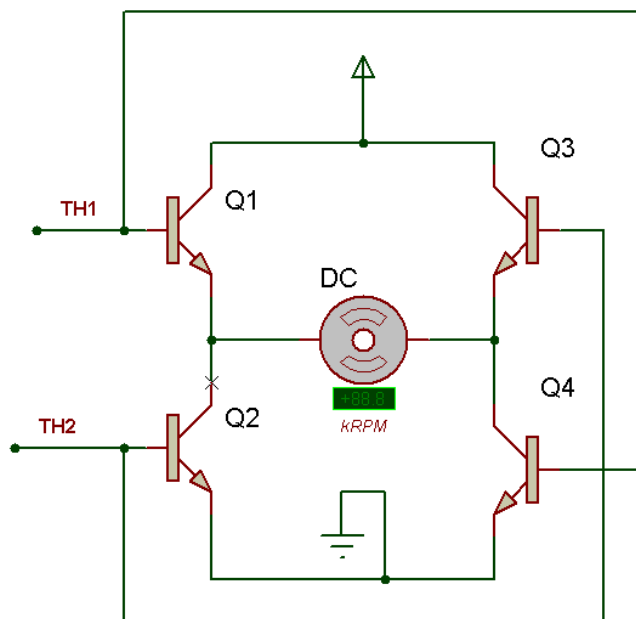
1. Sơ đồ nguyên lý.



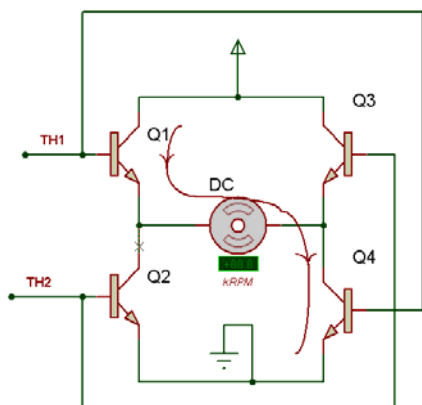
Mạch cầu H dùng để đảo chiều động cơ 1 chiều 1 cách thuận tiện.

Để giải thích nguyên lý hoạt động ta lấy ví dụ mạch cầu H đơn giản cấu tạo bởi 4 transistor như sau:

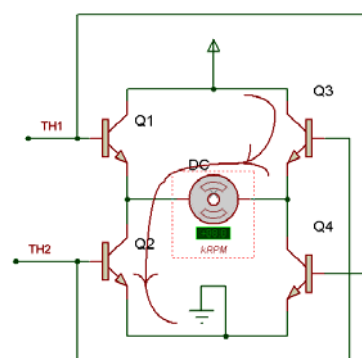
Sơ đồ gồm 4 van công suất được đấu nối với nhau thành mạch cầu và có 2 tín hiệu điều khiển TH1 và TH2.



Nguyên lý hoạt động: Để động cơ quay thuận hay quay ngược thì ta sẽ cấp các tín hiệu TH1 và TH2 để đóng mở các van tương ứng. Cụ thể như sau:

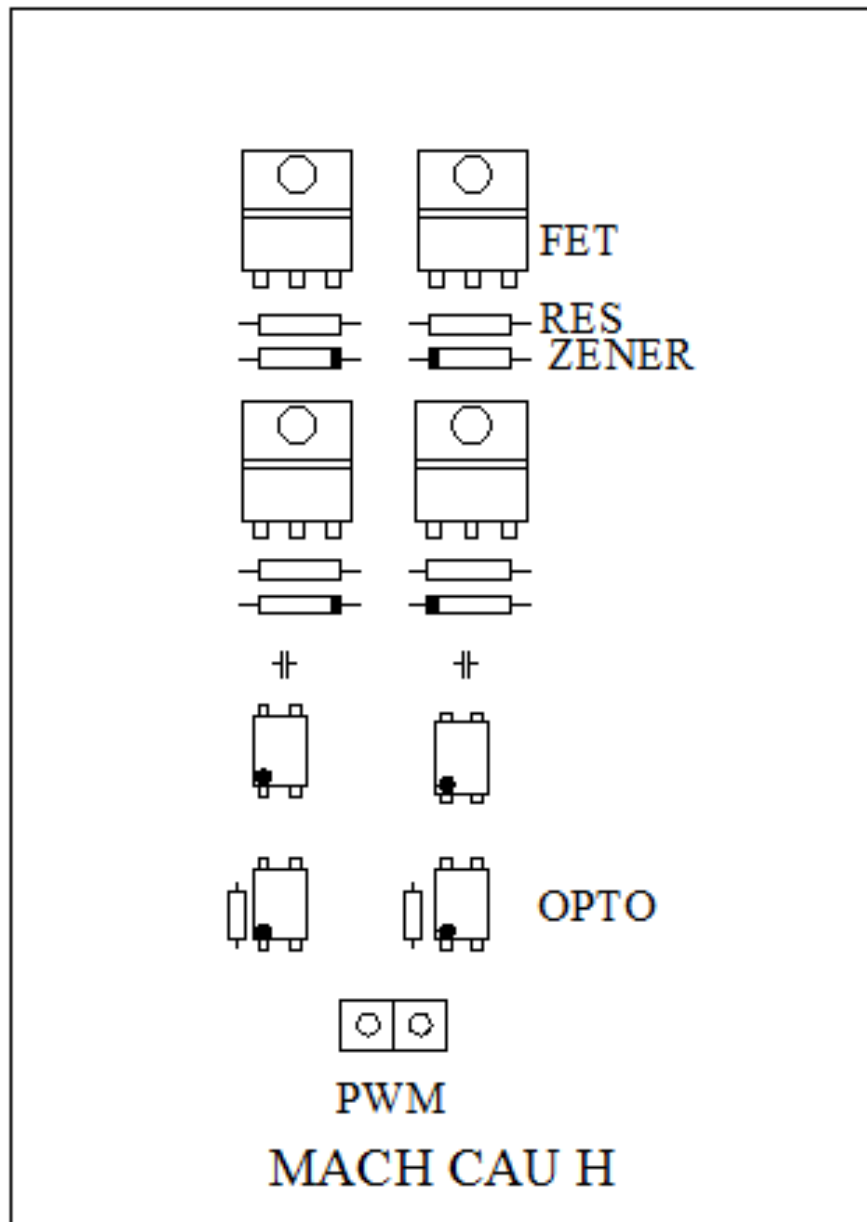


Cấp tín hiệu TH1 để các val Q1, Q4 là dẫn và Q2, Q3 là khóa. Động cơ quay thuận.



Cấp tín hiệu TH2 để các val Q2, Q3 là dẫn và Q1, Q4 là khóa. Động cơ quay ngược..

2. Mạch trên kit.



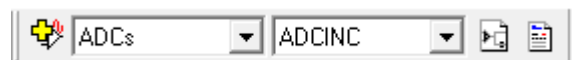
3. Nối cáp.

Nối chân P0.7 với 1 Jum PWM để điều khiển mạch cầu H.

4. Code chương trình.

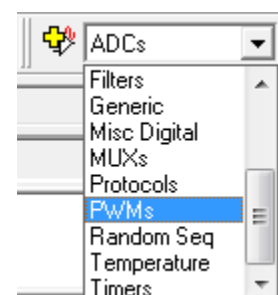
Thiết lập phần cứng.

Ta chọn các khối cần dùng trong

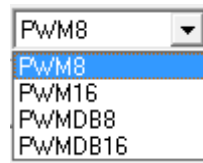


Trong bài này dùng PWM 8 bit

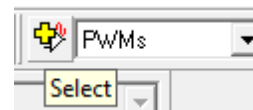
Chọn khối PWM bằng cách click vào biểu tượng và chọn khối PWMs.



Chọn PWM 8 bit ở mục



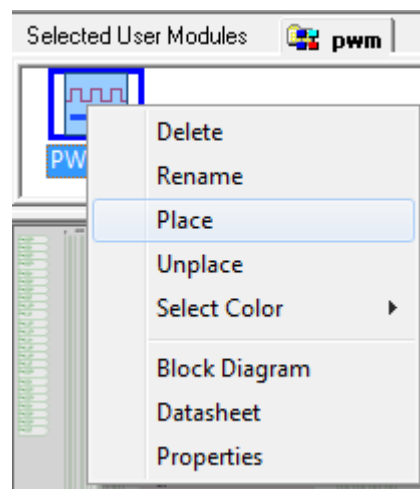
Chọn biểu tượng có dấu cộng



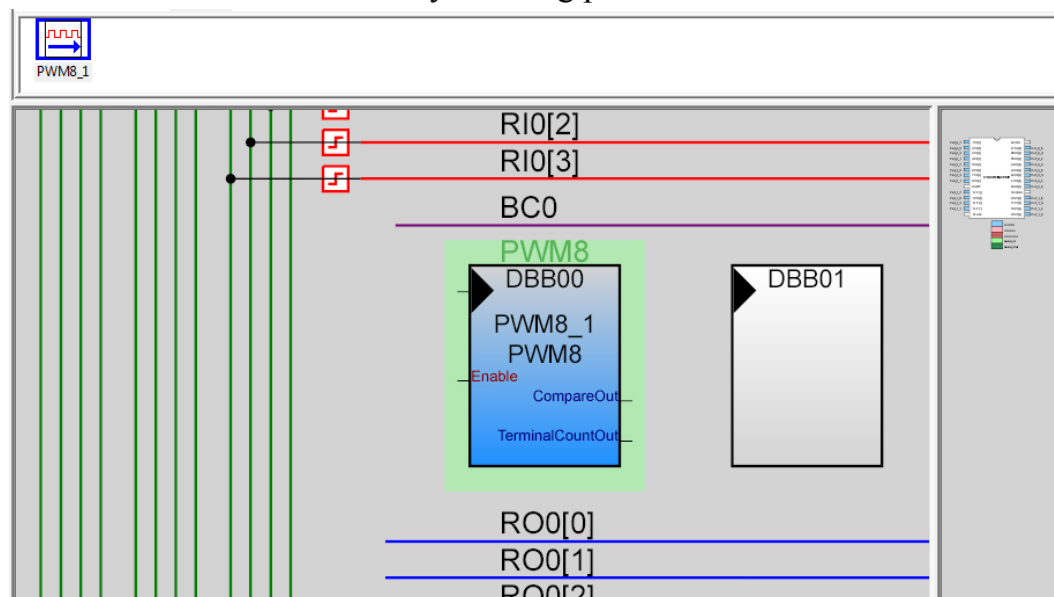
Khi đó khối PWM sẽ được chuyển xuống danh mục các khối đã được chọn.



Nháy chuột phải vào khối và chọn Place



Khi đó khối Pwm sẽ được chuyển xuống phía dưới.



Để phóng to giữ phím Ctrl và click chuột

Để thu nhỏ giữ phím Ctrl + Shift và click chuột.

Chọn clock cho Pwm.

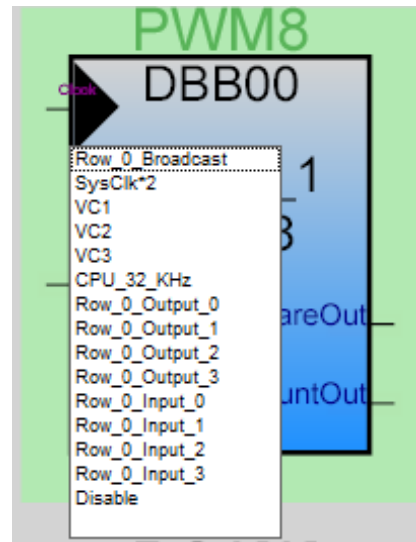
Sys = Clock*2:

Fclock = 24.2 = 48MHz.

VC1: Flock = 24MHz chia cho tỷ số chia chọn ở VC1.

Tương tự cho VC2 và VC3.

Row_0_input_0 chọn clock lấy từ bên ngoài...

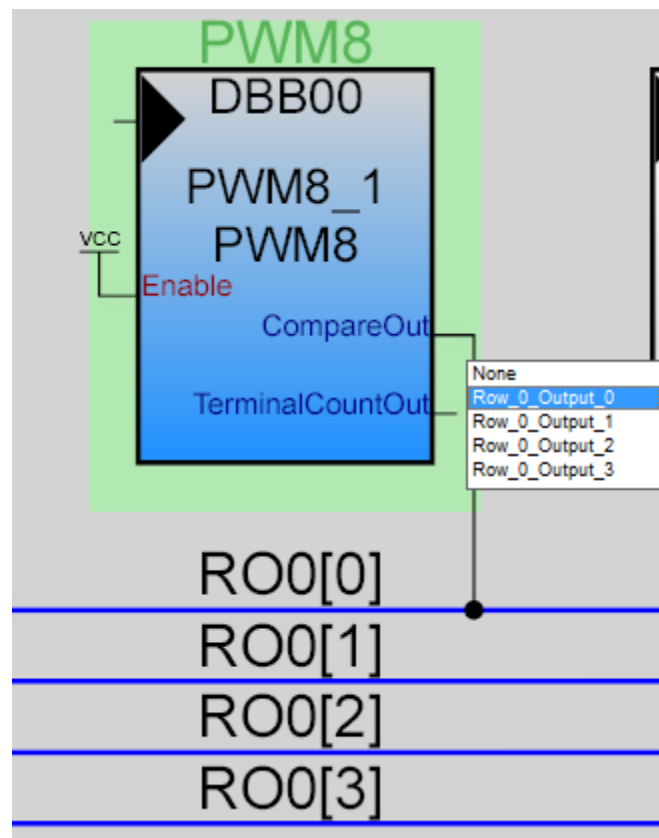


Chân Enable chọn là High để Pwm được phép hoạt động.

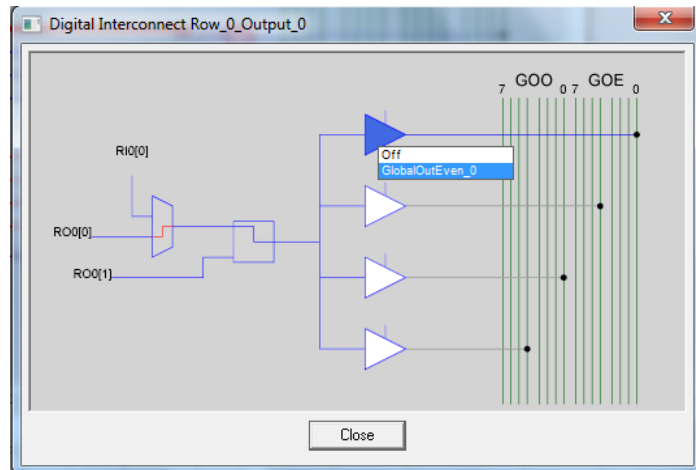
Đầu CompareOut để nối với 1 chân chip để đưa Pwm ra chân đẩy.

- Các chân làm đầu ra sẽ được nối với các hàng màu xanh (Vd RO0[0].)

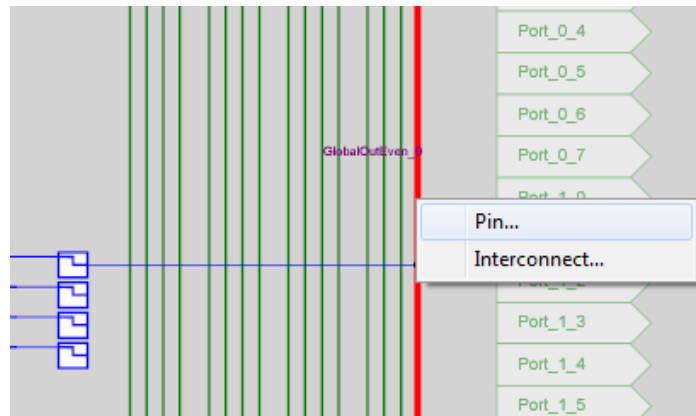
Chọn chuột trái vào CompareOut và chọn các hàng tương ứng. Sau khi chọn xong các dây sẽ được nối lại với nhau.



Tiếp theo các hàng sẽ được nối với các cột bằng cách click vào biểu tượng cuối của các hàng và chọn kết nối với 1 cột bất kỳ theo ý muốn.



Nổi các cột với các chân chip bằng cách click vào cột vừa chọn nổi dây và chọn Pin. Sau Đẩy nổi với pin mà mình mong muốn.



Chú ý: Các hàng màu xanh (RO) và các cột, các Pin bên trái là dành cho đầu ra. Các hàng màu đỏ, Cột bên phải, Pin bên phải là dành cho đầu vào.

Thiết lập tần số cho Pwm theo công thức sau:

$$F_{out} = \frac{Fin}{Thanh ghi + 1}.$$

Giá trị của thanh ghi sẽ được đặt trong thanh ghi Period trong câu lệnh:

```
PWM8 1 WritePeriod( 50);
```

Ta sẽ điều chỉnh giá trị Fin và giá trị thanh ghi để được tần số Pwm mong muốn.

Ví dụ muốn tạo ra tần số Pwm là 1khz.

Chọn tần số clock in là 100Khz.

$$\text{Suy ra Thanh ghi } +1 = \frac{\text{Fin}}{\text{Four}} = 100.$$

Vậy cần đặt cho thanh ghi giá trị là 99.

Tần số clock cho bộ Pwm có thể lấy từ ngoài chip, hoặc lấy từ thạch anh trong chip với các tỷ số chia khác nhau.

Power Setting: chọn tần số và điện áp hoạt động.
CPU_Clock: số kỳ thạch anh ứng với 1 lệnh.

VC1: Chọn tỷ số chia 1

VC2: Chọn tỷ số chia 2

VC3 Source: nguồn clock cho VC3.

VC3: Chọn tỷ số chia 3

Global Resources	Value
Power Setting [Vcc / SysClk freq	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	15
VC3 Source	VC2
VC3 Divider	10
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	[Vdd/2]+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
PWM8_1	

Với chip Psoc 24966 thì thạch anh trong là 20Mhz. Ta cần tần số clock cho bộ Pwm là 100 kHz. Vậy ta sẽ chọn các tỷ số chia như sau:

VC1 = 16, VC2 = 15, VC3 Source là VC2, VC3 = 10.

Và chọn Fclock cho Pwm là VC3 thì tần số clock cho Pwm sẽ là:

$$F_{clock} = V_{c3} = \frac{24MHz}{16.15.10} = 100kHz.$$

Một số thiết lập khác trong mục User Modul Parameters

Clock: Xung Clock.

Enable: cho phép hoạt động hay không.

Period giá trị thanh ghi định tần số.

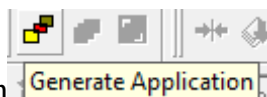
PulseWith: Độ rộng xung.

Copare Type chọn Less than.

ClockSync chọn là Sync to SysClk

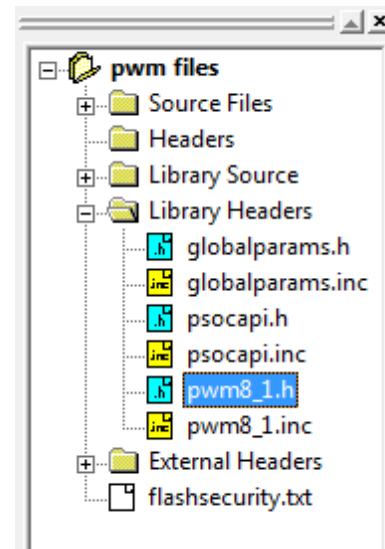
User Module Parameters	Value
Clock	VC3
Enable	High
CompareOut	Row_0_Output_1
TerminalCountOut	None
Period	99
PulseWidth	50
CompareType	Less Than
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

Các giá trị Period và PulseWith ta có thể đặt tại đây hoặc theo câu lệnh trong chương trình.



Sau đó Generate Application để xác lập cấu hình. Rồi chuyển sang giao diện viết chương trình.

Ta có thể xem các lệnh về Pwm trong thư mục Library Headers – Pwm8_1.h.



Một số lệnh cơ bản:

```
PWM8_1_EnableInt(); // Cho phép Pwm hoạt động
PWM8_1_DisableInt(); // Cấm
PWM8_1_Start();      // Bắt đầu chạy
PWM8_1_Stop();       // Dừng lại
PWM8_1_WritePeriod(); // Giá trị đặt tần số Pwm
PWM8_1_WritePulseWidth(); // độ rộng xung < = giá trị đặt trong Period.
```

Chương trình:

```
// 15.06.2011
// PWM điều chỉnh tốc độ động cơ qua mạch cầu H.
// Chan P1.7 nối với chân Điều khiển mô Fet.
#include <m8c.h> // part specific constants and macros
#include "PSoC_API.h" // PSoC API definitions for all User Modules
unsigned int time; // biến thời gian
unsigned char pwm; // biến thay đổi độ rộng xung

void int_sys()
{
    PWM8_1_WritePeriod(99); // giá trị thành ghi đưa vào để có
    T=1khz
    PWM8_1_WritePulseWidth(99); // PWM=0%
    PWM8_1_DisableInt();
    PWM8_1_Start();
}

void main()
{
    M8C_EnableGInt;
```

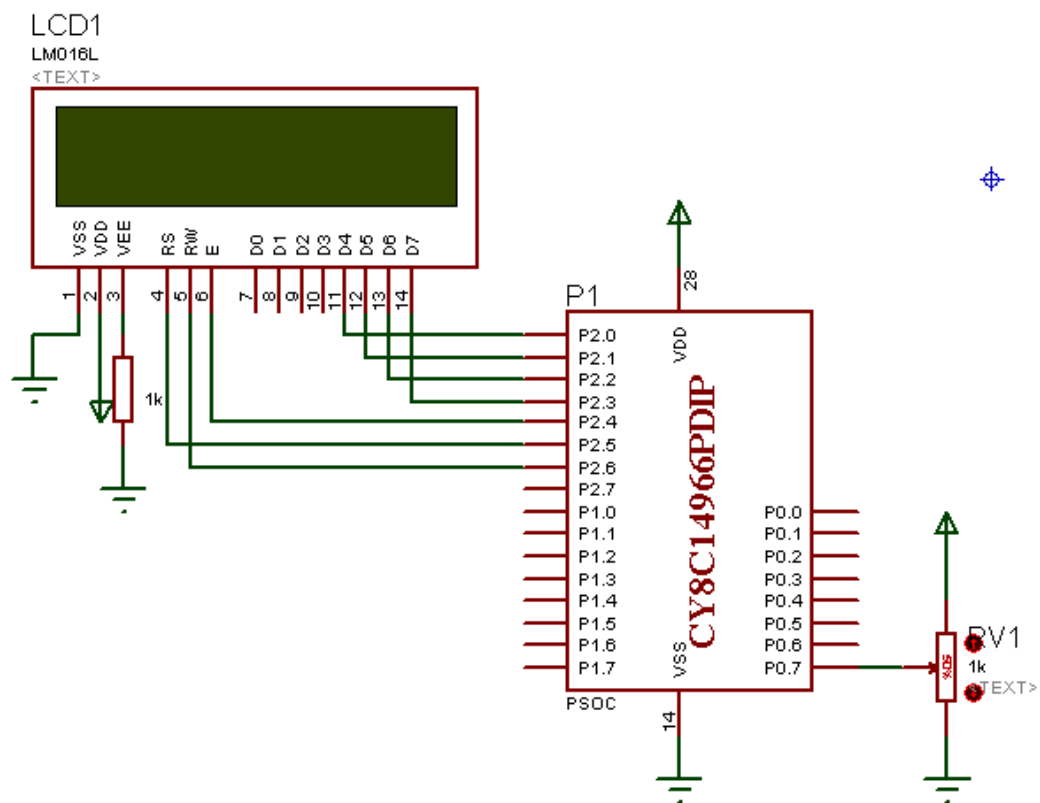
```

int_sys();
while(1)
{
    if(++time>10000)// thời gian thay doi dien ap
    {
        time=0;
        pwm++;
        if(pwm>99)
            pwm=99;
        PWM8_1_WritePulseWidth(99-pwm);
    }
}

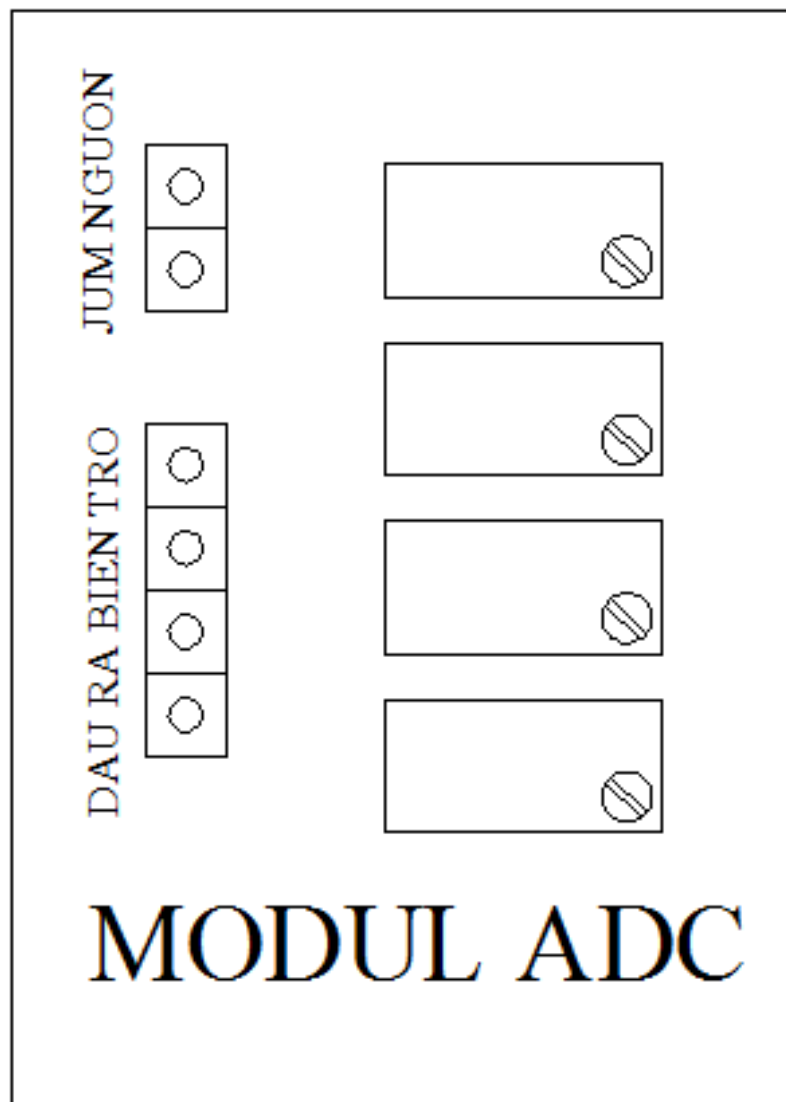
```

Bài 7. ADC

1. Sơ đồ nguyên lý



2. Mạch trên kit.



3. Nối cáp.

Port 2 nối với LCD

Chân P0.7 nối với biến trở.

4. Code chương trình.

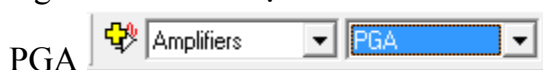
Thiết lập phần cứng:

Chọn khối ADC 8 bit trong mục ADCs ta chọn DELSIG8.



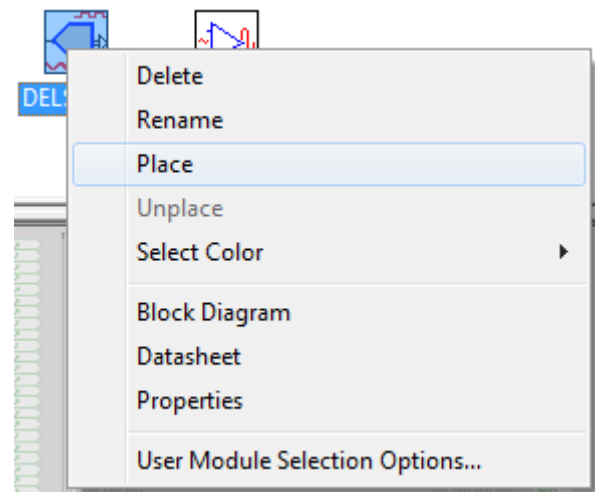
Và nhấn biểu tượng dấu cộng.

Ngoài ra ta cần chọn thêm khối khuếch đại trong mục Amplifiers chọn



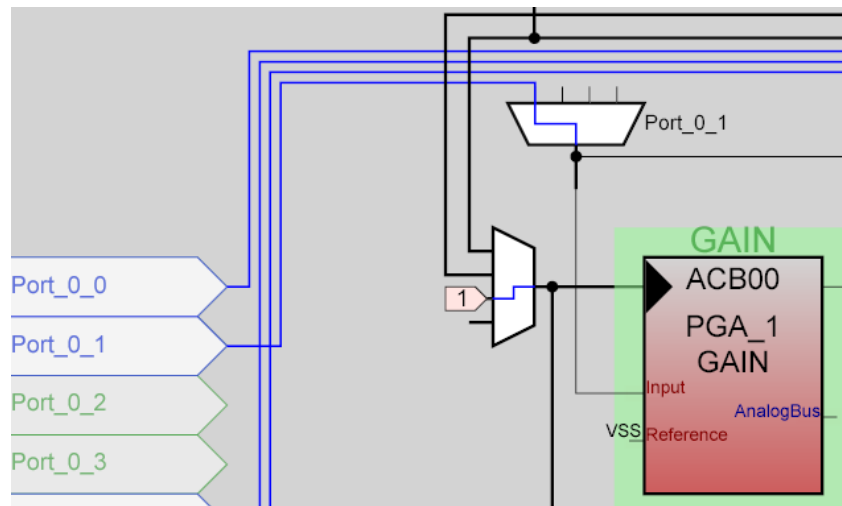
Sau đó nhấn vào biểu tượng dấu cộng bên cạnh để chuyển các bộ đã chọn xuống danh sách các khối đã chọn.

Đặt các khối xuống bằng cách chọn chuột trái và chọn Place.



Thiết lập cho bộ khuếch đại PGA:

Đầu vào input để chọn chân đưa tín hiệu từ triết áp vào.
Analogbus chọn none.
Reference chọn Vss.



Hệ số khuếch đại gain chọn 1.

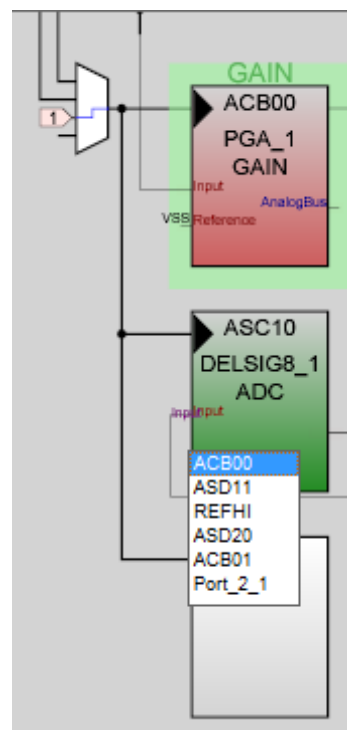
PGA_1	
User Module Parameters	Value
Gain	1.000
Input	AnalogColumn_InputMUX_0
Reference	VSS
AnalogBus	Disable

Trong mục Ref Mux chọn:
 $(V_{dd}/2) \pm (V_{dd}/2)$.
 Để chia đôi giải đầu vào làm 2 phần đối xứng bằng nhau.

Global Resources	Value
Power Setting [Vcc / SysClk freq	5.0V / 24MHz
CPU_Clock	SysClk/1
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	1
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	$(V_{dd}/2) \pm (V_{dd}/2)$
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

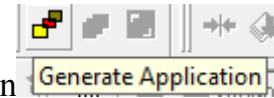
PGA_1

Chọn đầu vào input cho bộ adc là bộ khuếch đại: chọn ACB000 (địa chỉ bộ PGA).
 Chọn xung clock cho bộ ADC.



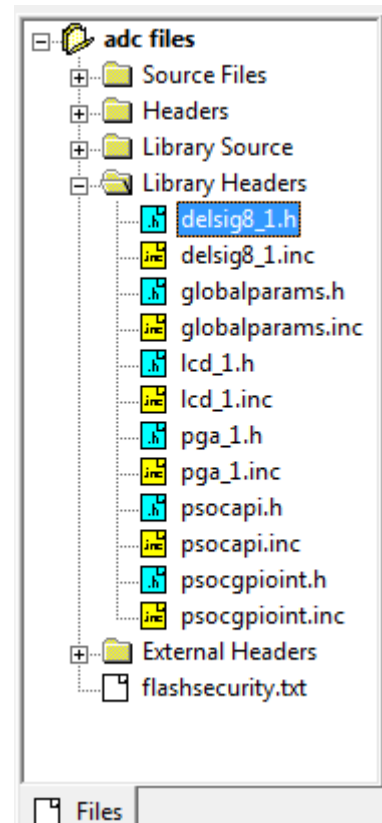
Để adc có thể hoạt động chọn Polling là Enable.

DELSIG8_1	
User Module Parameters	Value
TMR Clock	VC1
Input	ACB00
ClockPhase	Normal
Polling	Enable



Sau khi chọn các chân là Strong chọn Generate Application để xác lập cấu hình. Rồi chuyển sang giao diện viết chương trình.

Có thể lấy các lệnh của ADC trong mục như sau: *file – Library Headers – delsig8_1.h



Code chương trình.

```
//15.06.2011
```

```
// Do điện áp từ 0 - 5V Sử dụng bộ ADC
```

```
// Port 2 nối với LCD
```

```
// Chân P0.7 nối với biến trở.
```

```
#include <m8c.h> // part specific constants and macros
```

```
#include "PSoC_API.h" // PSoC API definitions for all User Modules
```

```
#include <math.h>
```

```
unsigned int adc, dienap;
```

```
void int_sys() // Khởi tạo
```

```
{
```

```
    LCD_1_Start();
```

```
DELSIG8_1_Start(DELSIG8_1_HIGHPOWER); //Khoi tao gia tri

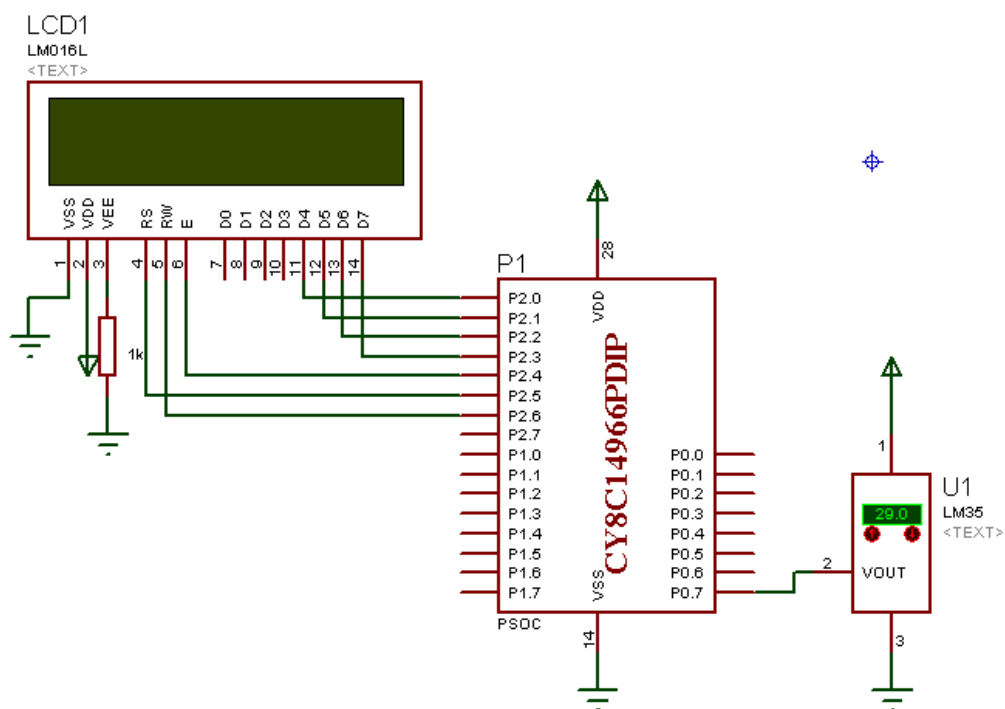
DELSIG8_1_StartAD(); // Bắt đầu chuyển đổi
DELSIG8_1_SetPower(DELSIG8_1_HIGHPOWER);
PGA_1_Start(PGA_1_HIGHPOWER);
}

void Doc_ADC()
{
    int d;
    while(!DELSIG8_1_fIsDataAvailable()); // cho bien doi xong
    d = DELSIG8_1_cGetData();
    DELSIG8_1_ClearFlag();
    adc = d + 128;
}

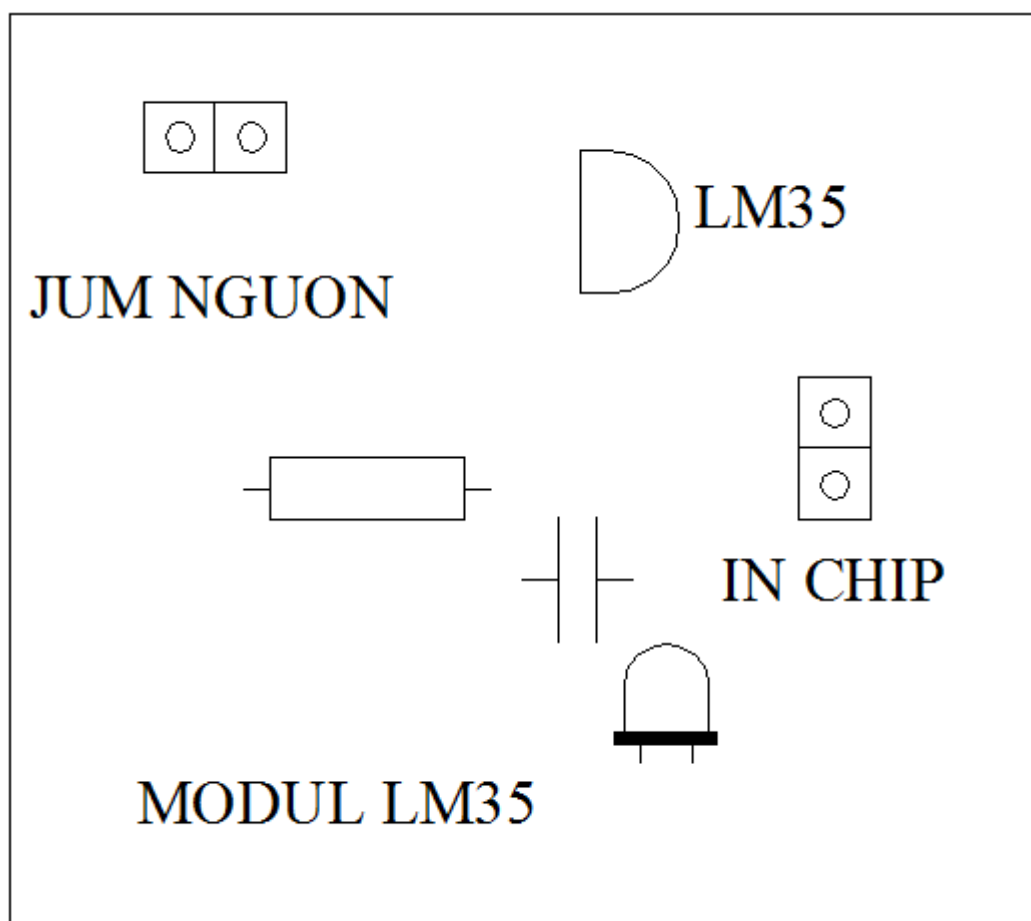
void main()
{
    M8C_EnableGInt; // cho phép ngắt toàn cục
    int_sys();
    while(1)
    {
        Doc_ADC();
        dienap = adc*0.02; // 0.22 = 5/(2^11)
        LCD_1_Position(0,0);
        LCD_1_PrCString("Dien Ap: ");
        LCD_1_WriteData('0'+dienap);
        LCD_1_PrCString(" Vol");
    }
}
```

Bài 8. LM35

1. Sơ đồ nguyên lý.



2. Mạch trên kit.



3. Nối cáp.

Chan P0_7 nối với LM35. Để Pull Up.

P2 dùng cho LCD.

4. Code chương trình.

Trong modul này sử dụng LM35 để đo nhiệt độ.

LM35 sẽ thay đổi điện áp ở ngõ ra tương ứng với nhiệt độ môi trường. Vì vậy cần sử dụng bộ adc để đọc điện áp ở đầu ra của LM35 và quy đổi ra nhiệt độ. Và LCD để hiển thị.

Dùng ADC 10 bit, sensor LM35 (10 mV / 1 độ C) Xác định nhiệt độ đo được qua số đo trên chân ADC: Ta có: 5000 mV --- ứng với --- 1023 (thang đo ADC 10 bit) Vậy: 10 mV --- ứng với --- $10 \times 1023 / 5000 = 2,046$ Con số 2,046 tính được đó chính là lượng thay đổi trên chân ADC ứng với thay đổi 10mV ở đầu ra LM35 hay ứng với thay đổi 1 độ C trên LM35. Suy ra nhiệt độ đo được: Nhiệt độ = $ADC_Read(0) / 2,046$ (độ C)

Thiết lập phần cấu hình cho bộ ADC như trong bài ADC.

// Đo nhiệt độ sử dụng LM35

```
#include <m8c.h> // part specific constants and macros
```

```
#include "PSoCAPI.h" // PSoC API definitions for all User Modules
```

```
#include <math.h>
```

```
unsigned int abc;
```

```
unsigned char t;
```

```
void sys()
```

```
{
```

```
    M8C_EnableGInt;
```

```
    LCD_1_Start();
```

```
    PGA_1_Start(PGA_1_HIGHPOWER);
```

```
    DELSIG11_1_SetPower(DELSIG11_1_HIGHPOWER);
```

```
    DELSIG11_1_StartAD();
```

```
}
```

```
void GetAD()
```

```
{
```

```
    int d;
```

```
    while(!DELSIG11_1_IsDataAvailable());
```

```
    d = DELSIG11_1_GetData();
```

```
    DELSIG11_1_ClearFlag();
```

```
    abc = d + 1024;
```

```
}
```

```
void main()
```

```
{
```

```
    sys();
```

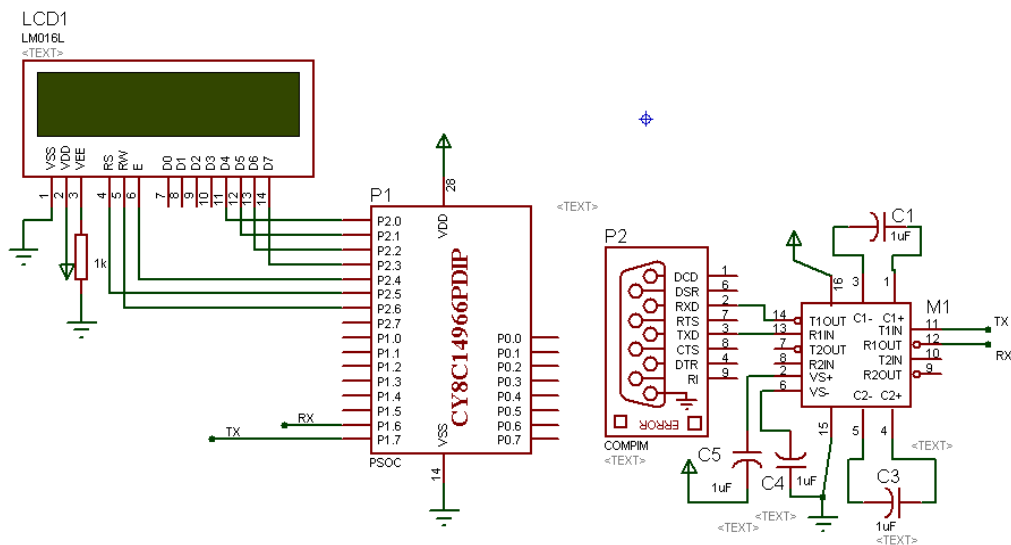
```

while(1)
{
    GetAD();
    t = 2+0.246*abc;
    LCD_1_Position(0,0);
    LCD_1_PrCString(" Nhiệt Do: ");
    LCD_1_WriteData('0'+t/10);
    LCD_1_WriteData('0'+t%10);
    LCD_1_Position(1,0);
    LCD_1_PrCString(" ADC: ");
    LCD_1_WriteData('0'+abc/1000);
    LCD_1_WriteData('0'+abc/100%10);
    LCD_1_WriteData('0'+abc/10%10);
    LCD_1_WriteData('0'+abc%10);
}
}

```

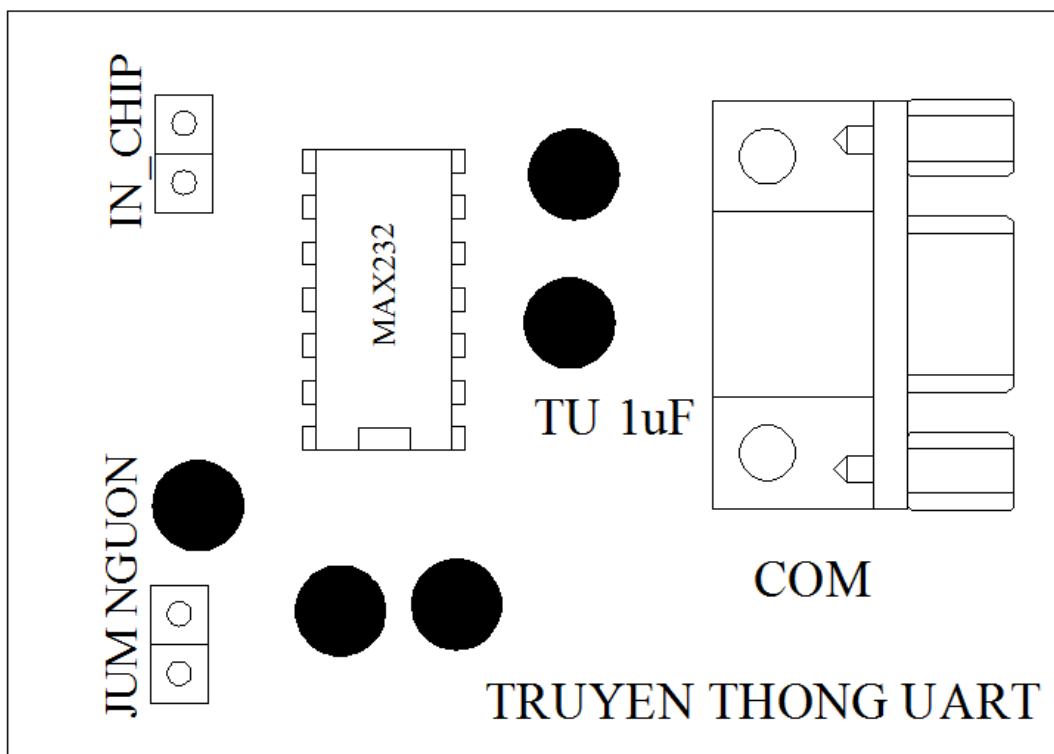
Bài 9. Truyền thông Uart.

1. Sơ đồ nguyên lý.



Chân 5 của cổng com được nối với GND.

2. Mạch trên kit.



3. Nối cáp.

Port 2 nối với LCD để hiển thị.

Chân P1.6(RX của chip) nối với chân TX của Max232, và chân P1.7 (chân TX của chip) nối với chân RX của max 232.

4. Code chương trình.

Bài này thực hiện truyền thông nối tiếp RS232 qua cổng com.

Cần sử dụng 1 bộ LCD để hiển thị.

1 bộ Counter để chia tần.

1 bộ UART(gồm bộ Rx và Tx) để thực hiện truyền thông.

Tính tốc độ truyền: Baud.

$$\text{Baud} = \frac{\text{Fclock}}{(\text{Thanh ghi Counter} + 1) \times 8}$$

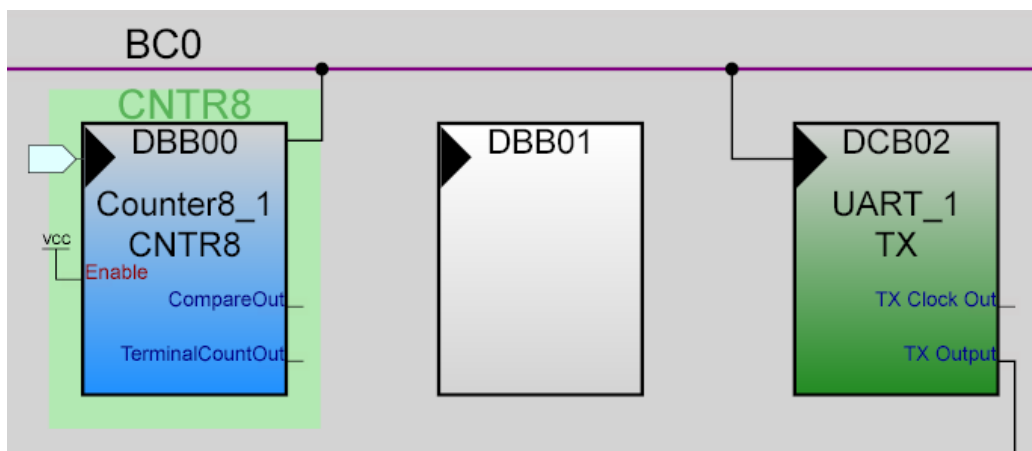
Fclock: xung kích đưa vào bộ Counter để chia tần.

Ví dụ truyền với tốc độ 38400 baud, Fclock = sys*2 = 48MHz.

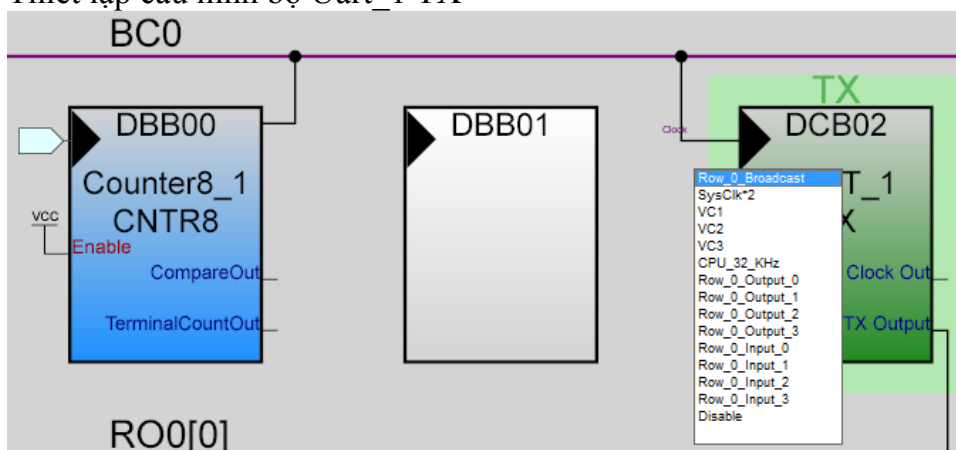
$$\text{Suy ra Thanh ghi Counter} = \frac{48\text{MHz}}{(38400) \times 8} - 1 = 155.$$

Thiết lập cấu hình bộ counter:

Counter8_1	
User Module Parameters	Value
Clock	SysClk*2
ClockSync	Sync to SysClk
Enable	High
CompareOut	None
TerminalCountOut	None
Period	155
CompareValue	78
CompareType	Less Than
InterruptType	Terminal Count



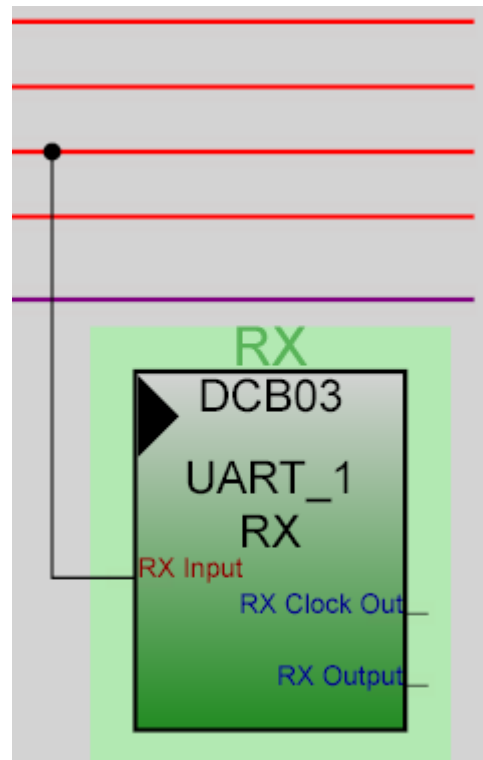
Thiết lập cấu hình bộ Uart_1 TX



Clock lấy từ bộ Counter được nối với nhau thông qua dây BC0.

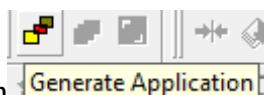
Đầu ra TX Out nối với chân P1.7

Tương tự cho RX.
Rx Input được nối với các hàng màu đỏ và nối với các cột bên trái rồi ra chân P1.6



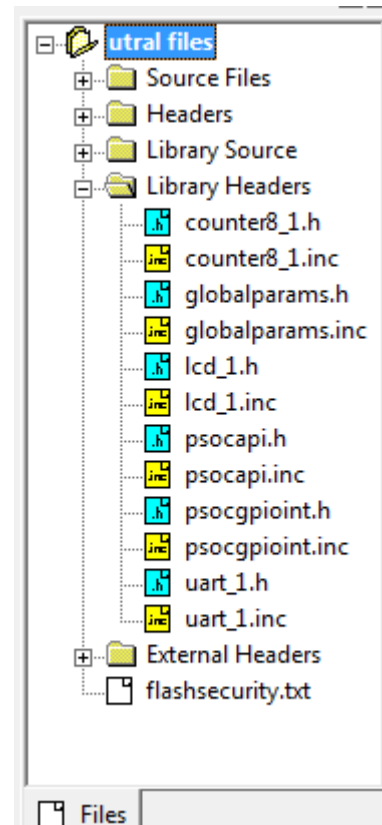
Thiết lập cấu hình UART_1.

User Module Parameters	Value
Clock	Row_0_Broadcast
RX Input	Row_0_Input_2
TX Output	Row_0_Output_3
TX Interrupt Mode	TXRegEmpty
ClockSync	Sync to SysClk
RxCmdBuffer	Enable
RxBufferSize	16
CommandTerminator	13
Param_Delimiter	32
IgnoreCharsBelow	32
Enable_BackSpace	Disable
RX Output	None
RX Clock Out	None
TX Clock Out	None
InvertRX Input	Normal



Sau đó Generate Application để xác lập cấu hình. Rồi chuyển sang giao diện viết chương trình.

Xem các lệnh về các bộ đã chọn trong mục:
*file – Library Headers - *.h



Chương trình.

```
// Tx P1.7
// Rx P1.6
#include <m8c.h>      // part specific constants and macros
#include "PSoCAPI.h"  // PSoC API definitions for all User Modules
void int_sys()
{
    UART_1_Start(UART_PARITY_NONE);
    Counter8_1_Start();
    LCD_1_Start();
}
unsigned char data;
unsigned int time;
void main()
{
    M8C_EnableGInt;
    int_sys();
    while(1)
    {
        if(++time>20000)
            // truyền kí tự 12 lên máy tính sau mỗi khoảng thời gian time
    }
}
```

```

    {
        time=0;
        UART_1_PutChar('0'+1);
        UART_1_PutChar('0'+2);
    }
    if((UART_1_bReadRxStatus() & UART_1_RX_REG_FULL)!=0)
// khi nhan 1 ky tu tu ban phim
    {
        data = UART_1_bReadRxData();

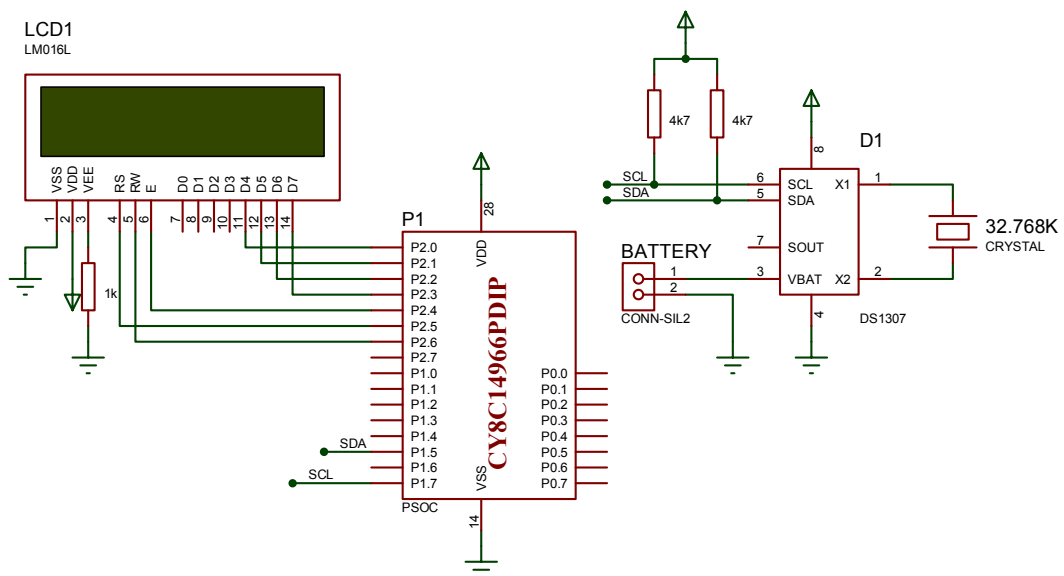
// nhan du lieu tu may tinh
    }
    LCD_1_Position(0,0);
    LCD_1_PrCString("Ky Tu Nhan: ");
    LCD_1_Position(1,0);
    LCD_1_WriteData(data); // Hien thi ki tu nhan duoc len LCD

}
}

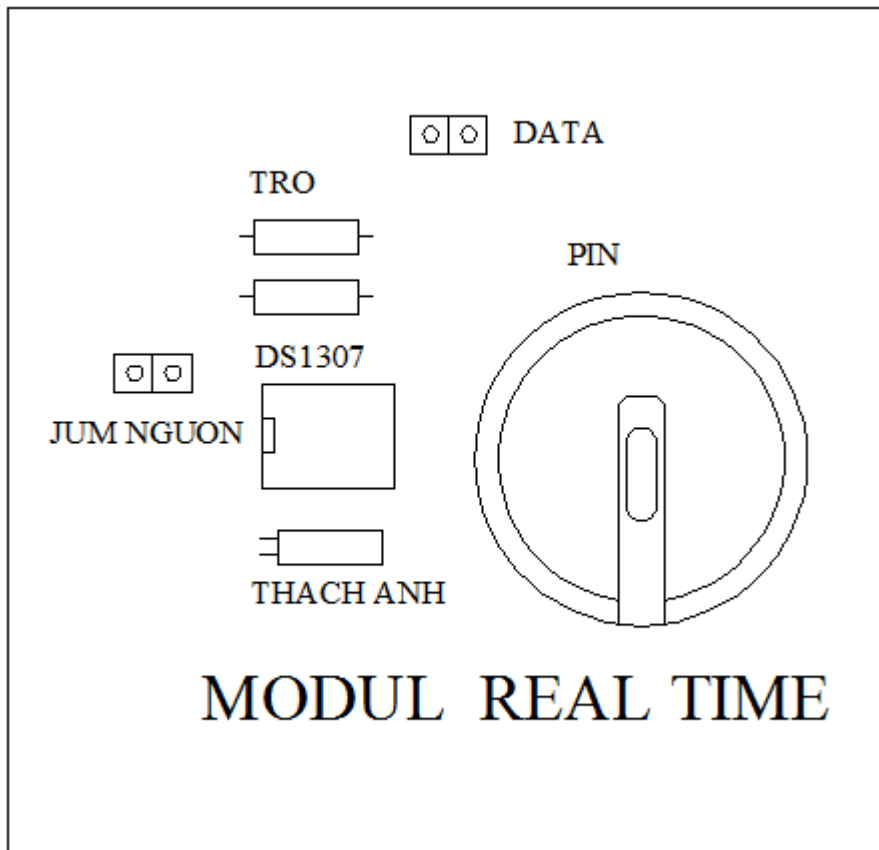
```

Bài 10. DS1307.

1. Sơ đồ nguyên lý.



2. Mạch trên kit.



3. Nối cáp.

Nối chân P1.5 và chân P1.7 với chân data và cắm jum nguồn cho modul.

4. Code chương trình.

```
#include <m8c.h> // part specific constants and macros
#include "PSoC_API.h" // PSoC API definitions for all User Modules
#include "stdlib.h"
#define mode (PRT0DR&0x80)
#define up (PRT0DR&0x20)
#define down (PRT0DR&0x08)
int sec,min=59,hour=23,week=7,day=28,month=2,year=10;
unsigned char chedo,kieugio=0;
BYTE tData[9]={0,0,0,0,0,0,0,0,0x93}; // bat dau = 0 va ket thuc bang 0x93 de
//ghi vao thanh ghi DK RTC
BYTE rData[8];
void delay()
{
    unsigned int i,j;
    for(i=0;i<3000;i++);
    for(j=0;j<40000;j++);
}
```

```

void StartSystem()
{
    M8C_EnableGInt;
    LCD_1_Start();
    I2CHW_1_Start();
    I2CHW_1_EnableMstr();
    I2CHW_1_EnableInt();

}

void LCDdisplay(int Data)
{
    LCD_1_WriteData('0'+Data/10);
    Data=Data%10;
    LCD_1_WriteData('0'+Data);
}

void LCD_clear()
{
    LCD_1_Position(0,0);
    LCD_1_PrCString("          ");
    LCD_1_Position(1,0);
    LCD_1_PrCString("          ");
}

void ReadRTC()
{
    I2CHW_1_bWriteBytes(0x68,tData,1,I2CHW_1_NoStop);
                                // Dung viet du lieu
    I2CHW_1_ClrWrStatus();
    I2CHW_1_fReadBytes(0x68,rData,7,I2CHW_1_RepStart);
    while(!(I2CHW_1_bReadI2CStatus()&I2CHW_RD_COMPLETE));
    I2CHW_1_ClrRdStatus();
    sec=BCD_conver_DEC(rData[0]&0x7F);
    min=BCD_conver_DEC(rData[1]);
    if(1)
        hour=BCD_conver_DEC(rData[2]&0x3F); // che do 12h
    else
        hour=BCD_conver_DEC(rData[2]&0x1F); // che do 24h
    week=BCD_conver_DEC(rData[3]);
    day=BCD_conver_DEC(rData[4]);
    month=BCD_conver_DEC(rData[5]);
    year=BCD_conver_DEC(rData[6]);
}

```

```
    }  
void Write_RTC()  
{  
    tData[0]=0;  
    tData[1]=DEC_conver_BCD(sec);  
    tData[2]=DEC_conver_BCD(min);  
    tData[3]=DEC_conver_BCD(hour); // che do 24h  
    tData[4]=DEC_conver_BCD(week);  
    tData[5]=DEC_conver_BCD(day);  
    tData[6]=DEC_conver_BCD(month);  
    tData[7]=DEC_conver_BCD(year);  
    tData[8]=0x93;  
    I2CHW_1_bWriteBytes(0x68,tData,9,I2CHW_1_CompleteXfer) ;  
  
    while(!I2CHW_1_bReadI2CStatus()I2CHW_WR_COMPLETE);  
  
    I2CHW_1_ClrWrStatus();  
}  
int BCD_conver_DEC(int BCD)  
{  
    int L,H;  
    L=BCD&0x0F;  
    H=(BCD>>4)*10;  
    return (H+L);  
}  
int DEC_conver_BCD(int DEC)  
{  
    int L,H;  
    L=DEC%10;  
    H=(DEC/10)<<4;  
    return (H+L);  
}  
void dislay_time()  
{  
    LCD_1_Position(0,12);  
    LCD_1_PrCString("20");  
    LCD_1_Position(0,6);  
    LCDdislay(day);  
    LCD_1_PrCString("-");  
    LCDdislay(month);  
    LCD_1_PrCString("-");  
    LCD_1_Position(0,14);  
    LCDdislay(year);
```

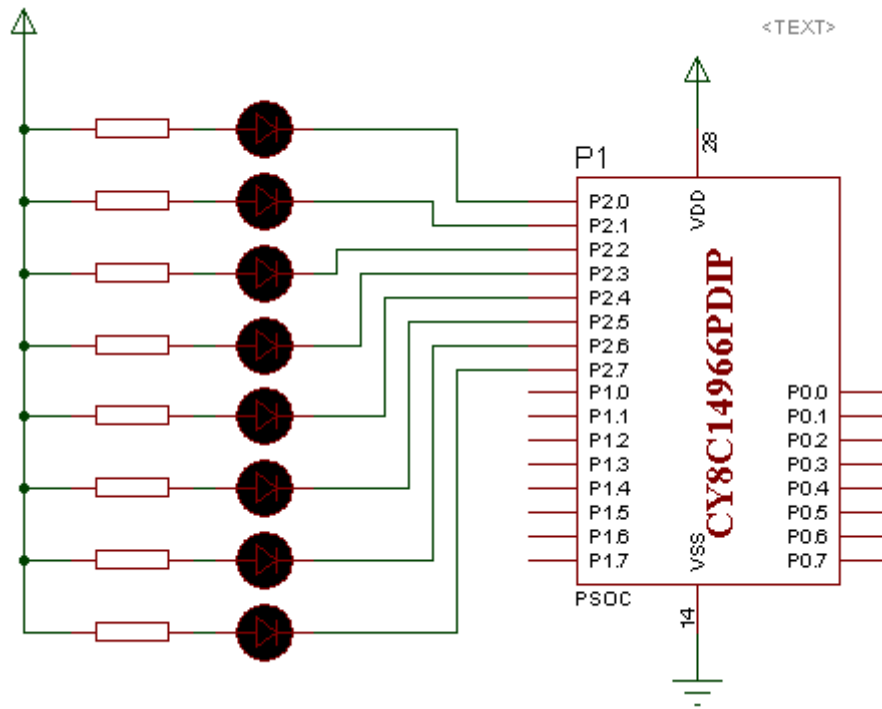
```
LCD_1_Position(1,3);
LCDdislay(hour);
LCD_1_PrCString(":");
LCDdislay(min);
LCD_1_PrCString(":");
LCDdislay(sec);
switch (week)
{
    case 1 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Mon");
        break;
    }
    case 2 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Tue");
        break;
    }
    case 3 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Wen");
        break;
    }
    case 4 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Thu");
        break;
    }
    case 5 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Fri");
        break;
    }
    case 6 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Sat");
        break;
    }
    case 7 : {
        LCD_1_Position(0,1);
        LCD_1_PrCString("Sun");
        break;
    }
}
```



```
        }  
    }  
}  
  
void main()  
{  
    int a=0;  
    StartSystem();  
    for(a=0;a<2000;a++)  
    {  
        LCD_1_Position(0,1);  
        LCD_1_PrCString("TBD - 48");  
        LCD_1_Position(1,1);  
        LCD_1_PrCString("PSoc - Ds1307");  
    }  
    LCD_clear();  
    PRT0DR=0xff;  
    while(1)  
    {  
        ReadRTC();  
        dislay_time();  
    }  
}
```

Bài 11. Timer.

1. Sơ đồ nguyên lý.



2. Đấu nối dây.

Sử dụng Port2 để nối với các Led đơn và cấu hình là Strong.

Thiết lập timer.

Chọn khối timer trong mục timer. 

Tính tần số cho timer.

Hoàn toàn tương tự như Pwm ta có công thức như sau:

$$F_{out} = \frac{F_{clock}}{Thanh ghi + 1}$$

Trong bài này ta cần tạo ra tần số là 1Hz, tức là Timer = 1s. Nhưng với timer8 bit thì không tạo ra được khoảng thời gian dài như thế. Ta sẽ tạo timer = 0,1s. Sau đó cứ 10 lần ngắt thì ta đảo trạng thái của led 1 lần thì sẽ có led nhấp nháy với tần số 1hz.

- Timer_1: 0.1s (10Hz). Chọn clock cho timer là Vc3..

Chọn tỷ số chia:

VC1: 16

VC2: 15

VC3: 100

$$\text{Suy ra } F_{\text{Clock}} = \frac{\text{Sys}}{\text{VC1.VC2.VC3}} = \frac{24\text{Mhz}}{16.15.100} = 1\text{Khz.}$$

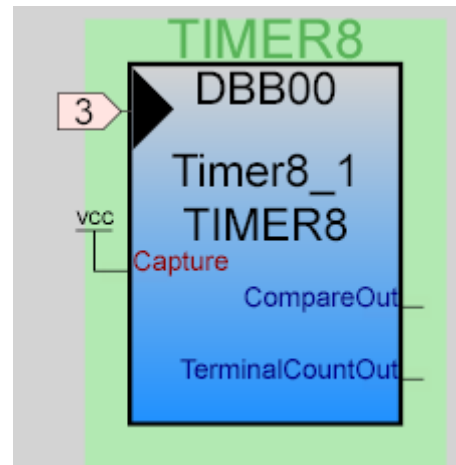
$$\text{Theo công thức ta có: Giá trị thanh ghi Timer1} + 1 = \frac{F_{\text{Clock}}}{F_{\text{out}}} = \frac{1\text{KHz}}{10\text{Hz}} = 100.$$

Vậy Giá trị thanh ghi Timer_1 = 99.

Chọn xung clock.

Capture chọn High để cho phép.

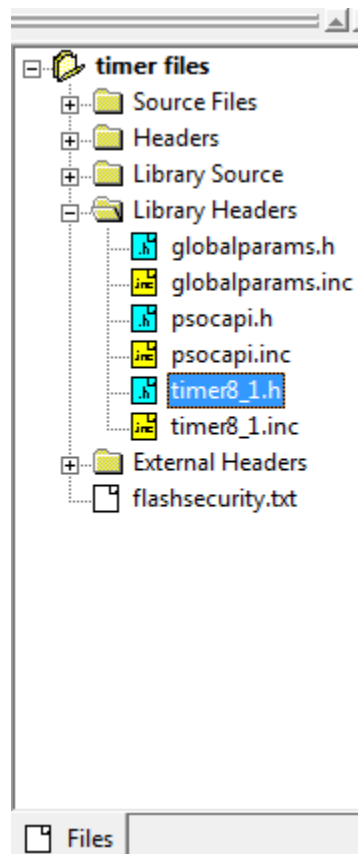
CompareOut, TerminalCountOut chọn None.



Timer8_1	
User Module Parameters	Value
Clock	VC3
Capture	High
TerminalCountOut	None
CompareOut	None
Period	99
CompareValue	0
CompareType	Less Than
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal

Sau đó application và chuyển sang viết chương trình:

Xem các lệnh về timer trong mục:
*file – Library Headres – Timer8_1.h



Một số lệnh timer:

```
Timer8_1_WritePeriod(99); // Gia tri thanh ghi dua vao timer
```

```
Timer8_1_WriteCompareValue(0); // khong so sanh
```

```
Timer8_1_EnableInt(); //cho phep ngat timer
```

```
Timer8_1_Start(); // Khoi tao module timer
```

Cấu trúc chương trình ngắt

```
#pragma interrupt_handler Timer8_1_ISR
```

```
void Timer8_1_ISR()
```

```
{
```

```
.....
```

```
}
```

```
void main()
```

```
{
```

```
while(1)
```

```
    {  
        ;  
    }  
}
```

3. Code chương trình.

// Nhảy led tần số 1 Hz.

```
#include <m8c.h> // part specific constants and macros
```

```
#include "PSoCAPI.h" // PSoC API definitions for all User Modules
```

```
unsigned char dem,;
```

```
void int_timer()
```

```
{  
    Timer8_1_WritePeriod(99); // timer=0.1s  
    Timer8_1_WriteCompareValue(0); // không so sánh  
    Timer8_1_EnableInt(); // cho phép ngắt time  
    Timer8_1_Start(); // Khởi tạo module time  
}
```

```
#pragma interrupt_handler Timer8_1_ISR
```

```
void Timer8_1_ISR()
```

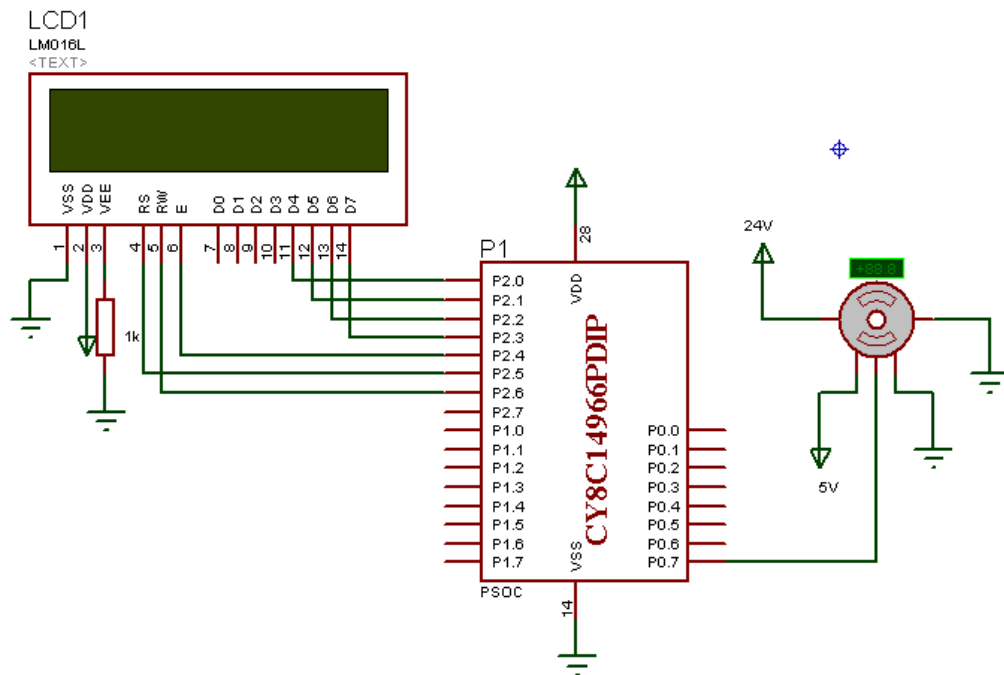
```
{  
    dem++;  
    if(dem>9)  
    {  
        dem = 0;  
        PRT2DR = ~PRT2DR;  
    }  
}
```

```
void main()
```

```
{  
    M8C_EnableGInt;  
    int_timer();  
    PRT2DR=0xff;  
    while(1)  
    {  
        ;  
    }  
}
```

Bài 12. Đo tốc độ động cơ (Timer, Counter).

1. Sơ đồ nguyên lý.



2. Nối cáp.

Port2 nối với LCD.

Chân P0.7 nối với đầu ra xung của encoder.

Dây đỏ của Encoder nối với 5V, dây đen nối với GND.

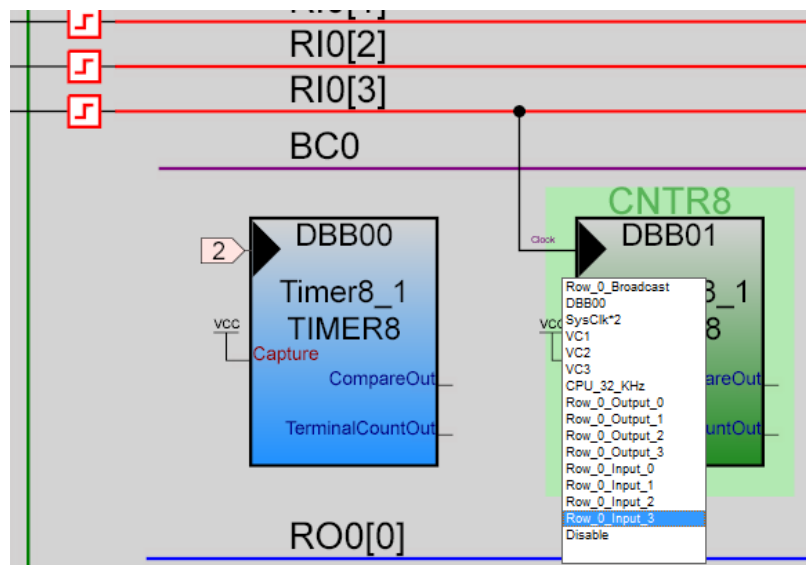
3. Code chương trình.

Trong bài này cần sử dụng khối LCD để hiển thị, Khối Counter, Timer để đếm xung đo tốc độ động cơ.

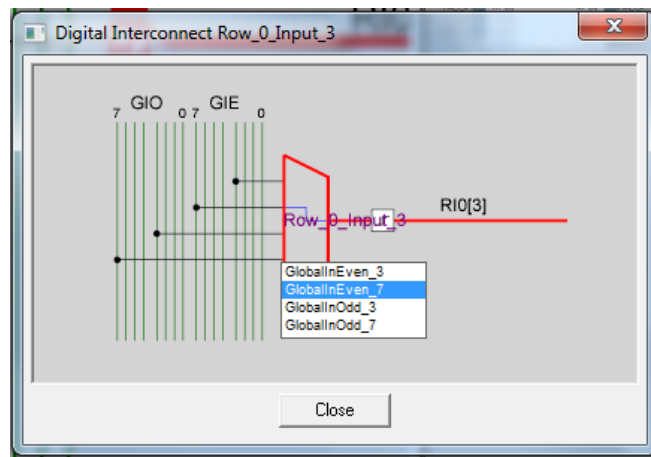
Thiết lập bộ counter: trong bài này bộ counter sử dụng để đếm số xung trả về từ encoder.

Clock cho counter lấy từ encoder. Chọn chuột trái và chọn

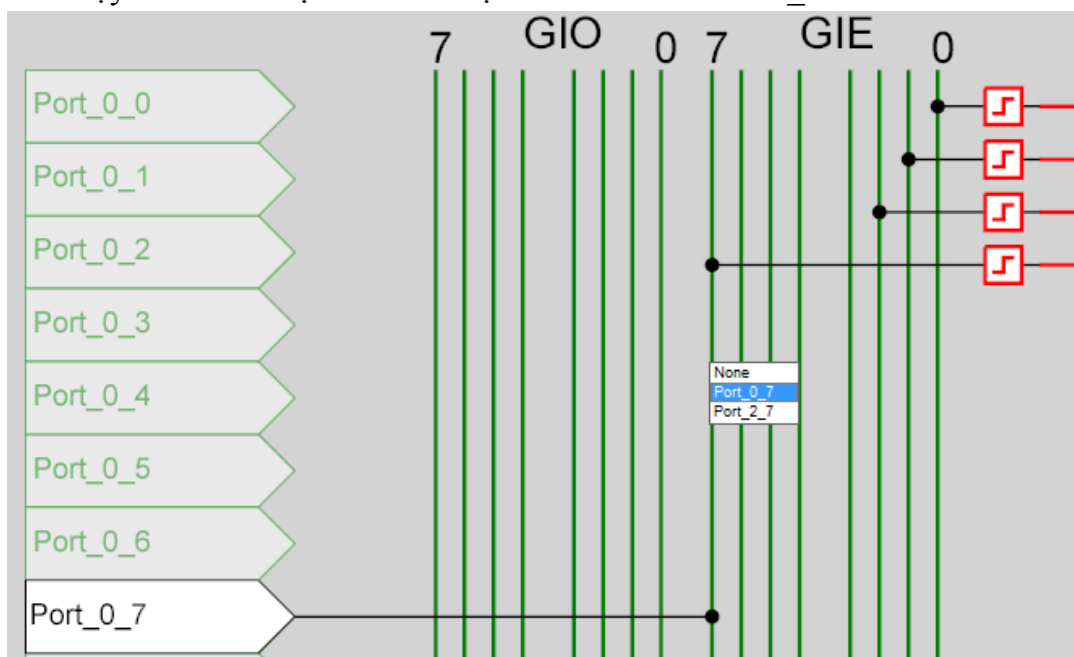
Row_0_Input_3 để nối dây từ counter nên RI0[3].



Từ RI0[3] chọn nối với GlobalEven_7 để có thể nối với P0.7



Tiếp theo chọn dây GIE_7 và chọn pin – chọn Pin 0_7.
Như vậy clock cho bộ Counter được đưa vào từ chân P0_7.



Các thiết lập khác:

Period = 255 thì
Counter sẽ đếm từ 255
về 0 rồi
Lặp lại

Counter8_1	
User Module Parameters	Value
Clock	Row_0_Input_3
ClockSync	Sync to SysClk
Enable	High
CompareOut	None
TerminalCountOut	None
Period	255
CompareValue	0
CompareType	Less Than
InterruptType	Terminal Count
InvertEnable	Normal

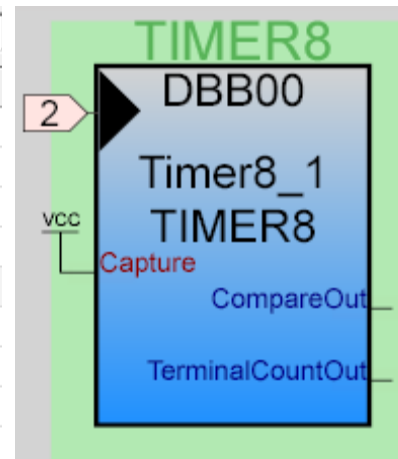
Thiết lập cấu hình bộ Timer: Tương tự như bài Timer. Chọn thời gian đếm xung trả về từ Encoder là 0.01s.

Tỷ số chia VC1 là 16, VC2 là 15, VC3 Source là VC2. VC3 là 10. Clock cho Timer là VC3.

Tout = Tin(Thanh ghi +1)

Suy ra: Thanh ghi = $\frac{T_{out}}{T_{in}} - 1 = \frac{F_{in}}{F_{out}} - 1 = \frac{24Mhz}{16.15.10} . 0,01 - 1 = 99$.

Timer8_1	
User Module Parameters	Value
Clock	VC2
Capture	High
TerminalCountOut	None
CompareOut	None
Period	0
CompareValue	0
CompareType	Less Than
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal



// Do tốc độ động cơ

// Encodo nối với chân P0.7

// LDC nối với Port2

#include <m8c.h> // part specific constants and macros

#include "PSoCAPI.h" // PSoC API definitions for all User Modules

unsigned old, new, count, v;

void delay(int t)

```
{
    int i,j;
```



```
        for(i = 0; i<t;i++)
            for(j = 0; j<100; j++);
    }
    void int_sys()
    {
        M8C_EnableGInt;
        LCD_1_Start();
        Timer8_1_WritePeriod(99); // time =0.01s
        Timer8_1_WriteCompareValue(0);
        Timer8_1_EnableInt(); // bat ngat timer
        Timer8_1_Start();

        Counter8_1_WritePeriod (255);
        Counter8_1_WriteCompareValue(0);
        Counter8_1_Start();
    }
    void lcd_2number(int Data)
    {
        LCD_1_WriteData('0'+Data/10);

        LCD_1_WriteData('0'+Data%10);
    }

#pragma interrupt_handler Timer8_1_ISR
void Timer8_1_ISR()
{
    old = new;
    new = 255 - Counter8_1_bReadCounter();// Couter dem giam
    if(old > new)
        count = 255 - old + new; // Su ly tran so
    else
        count = new - old; // Khong co tran so
}
void vantoc()
{
    v = count; // Do dong co tra ve 100 xung 1 vong
    LCD_1_Position(0,0);
    LCD_1_PrCString(" Van Toc: ");
    lcd_2number(v);
    LCD_1_Position(1,3);
    LCD_1_PrCString(" Vong/Giay ");
    delay(2000);
}
```

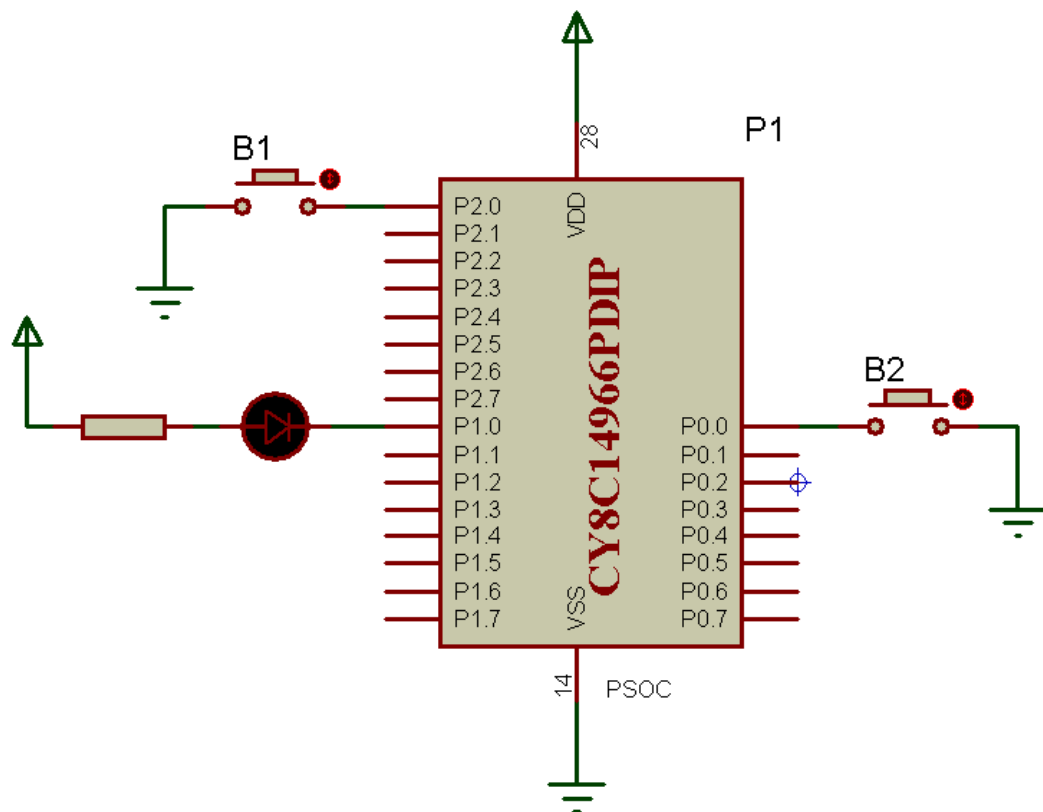
```

    }
void main()
{
    int_sys();
    while(1)
    {
        vantoc();
    }
}

```

Bài 13. Ngắt GPIO.

1. Sơ đồ nguyên lý.



Trong bài này sẽ thực hiện ngắt ngoài GPIO ở chân P0.0.

Ban đầu Led sẽ tắt, Khi bấm nút B2 thì sẽ xảy ra ngắt. Chương trình ngắt sẽ bật Led. Và khi nào bấm nút B1 sẽ tắt Led.



2. Thiết lập cấu hình.

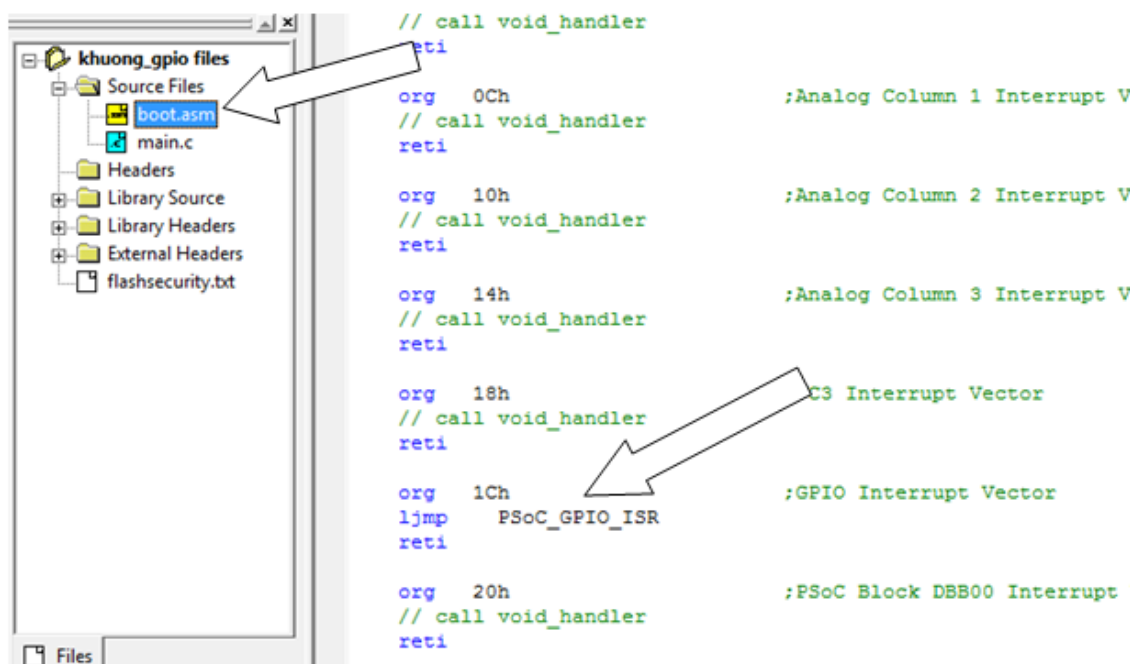
Sau khi tạo một Project mới ta sẽ chuyển sang thiết lập cấu hình và thiết lập như sau:

- Chân P1.0 nối với led để ở Strong.

- Chân P2.0 nối với nút bấm để ở Pull Up.
- Chân P0.0 nối với nút bấm để Pull Up trong mục Device và để FallingEde trong mục Interrupt để thực hiện ngắt khi có sườn xuống.

User Module Parameters		Value		
Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	Pull Up	DisableInt
Port_0_1	P0[1]	StdCPU	High Z Analog	DisableInt
Port_0_2	P0[2]	StdCPU	High Z Analog	FallingEdge
Port_0_3	P0[3]	StdCPU	High Z Analog	RisingEdge
Port_0_4	P0[4]	StdCPU	High Z Analog	ChangeFromRead
Port_0_5	P0[5]	StdCPU	High Z Analog	DisableInt
Port_0_6	P0[6]	StdCPU	High Z Analog	DisableInt
Port_0_7	P0[7]	StdCPU	High Z Analog	DisableInt
Port_1_0	P1[0]	StdCPU	Strong	DisableInt
Port_1_1	P1[1]	StdCPU	High Z Analog	DisableInt
Port_1_2	P1[2]	StdCPU	High Z Analog	DisableInt
Port_1_3	P1[3]	StdCPU	High Z Analog	DisableInt
Port_1_4	P1[4]	StdCPU	High Z Analog	DisableInt
Port_1_5	P1[5]	StdCPU	High Z Analog	DisableInt
Port_1_6	P1[6]	StdCPU	High Z Analog	DisableInt
Port_1_7	P1[7]	StdCPU	High Z Analog	DisableInt
Port_2_0	P2[0]	StdCPU	Pull Up	DisableInt
Port_2_1	P2[1]	StdCPU	High Z Analog	DisableInt
Port_2_2	P2[2]	StdCPU	High Z Analog	DisableInt

Sau đó click Geneate Application  và chuyển sang giao diện viết chương trình  :



Vào Source Files > boot.asm và tìm đến dòng:

```
org 1Ch      ;GPIO Interrupt Vector
ljmp  PSoC_GPIO_ISR
reti
```

Ta sẽ thay dòng PsoC_GPIO_ISR bằng _PsoC_GPIO_ISR (có thêm dấu _ phía trước) hoặc thành tên bất kỳ mà ta muốn.

Rồi nhấn Ctrl + S.

Chú ý:

- Phân biệt chữ hoa và chữ thường.
 - Sau mỗi lần Generate Application lại thì phải làm lại bước trên.
3. Code chương trình.

// 16.6.2011

```
#include <m8c.h>      // part specific constants and macros
#include "PSoC_API.h" // PSoC API definitions for all User Modules

// chương trình ngắt

#pragma interrupt_handler PSoC_GPIO_ISR

void PSoC_GPIO_ISR()
{
    PRT1DR = 0x00; // sang led
}

void main()
{
    M8C_EnableGInt; // cho phép ngắt
    M8C_EnableIntMask(INT_MSK0,0x20); // bật ngắt
    PRT0DR = 0xff;
```


ngắt cần có lệnh kiểm tra xem Pin nào bằng 0 khi ngắt xảy ra (tức là ngắt xảy ra trên pin đấy.).

Sau khi tạo một Project mới ta sẽ chuyển sang thiết lập cấu hình và thiết lập như sau:

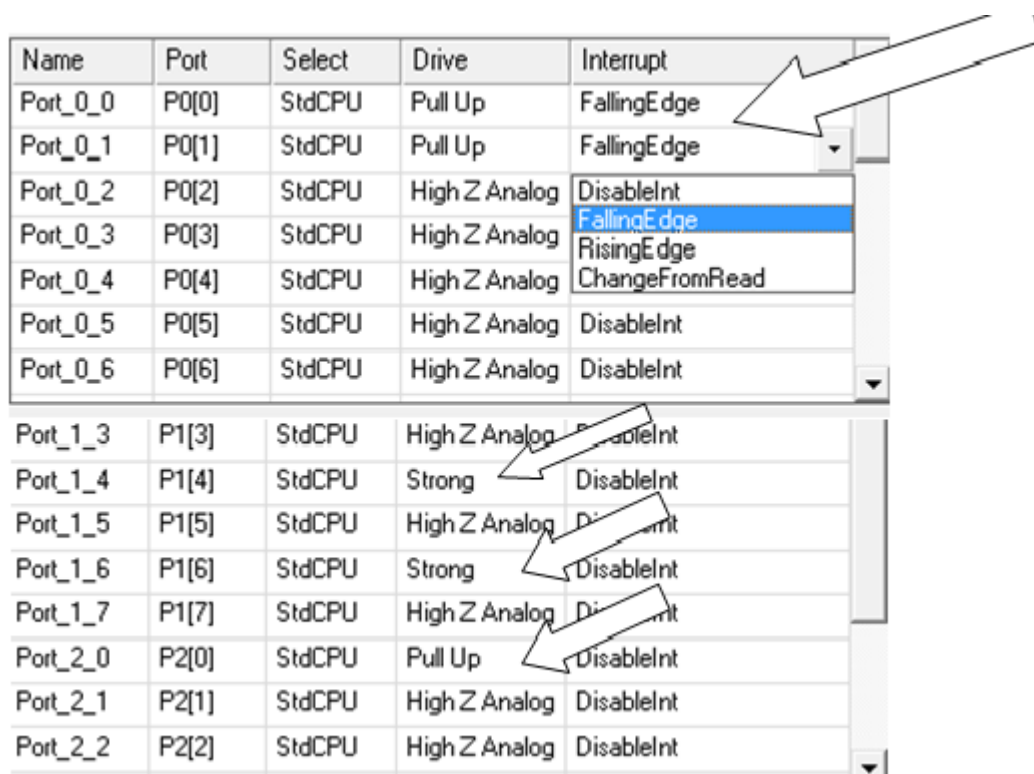
- Chân P1.4 nối với led D1 để ở Strong.

Chân P1.6 nối với led để D2 ở Strong.



- Chân P2.0 nối với nút bấm B3 để ở Pull Up.

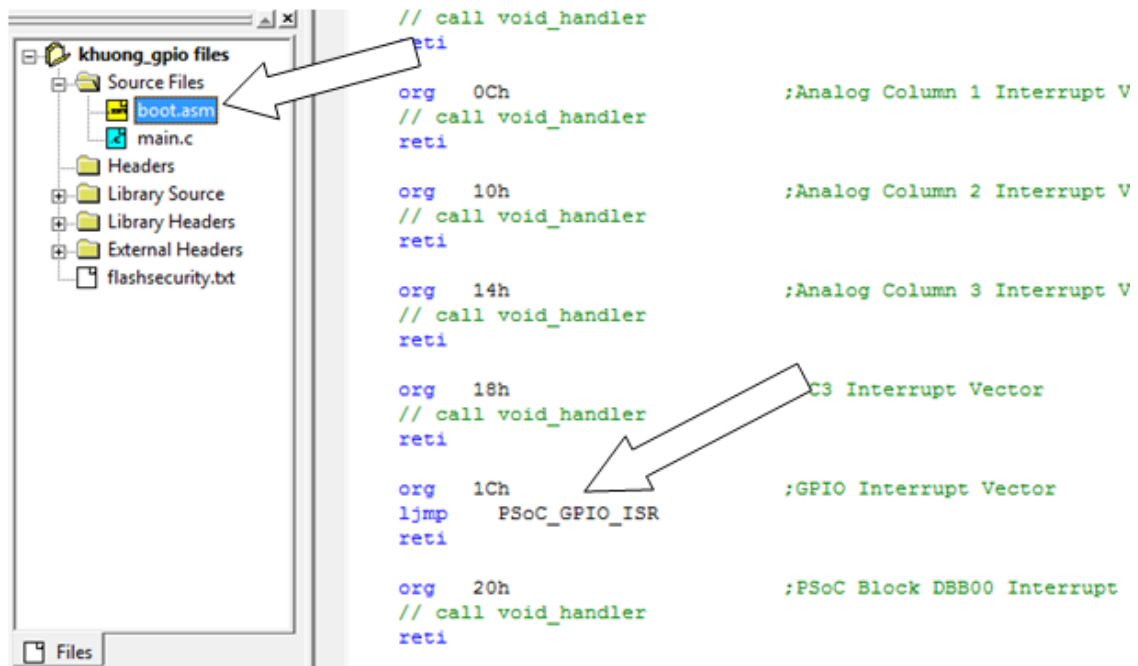
- Chân P0.0 nối với nút bấm B1 để Pull Up trong mục Device và để FallingEdge trong mục Interrupt để thực hiện ngắt khi có sườn xuống.

- Chân P0.1 nối với nút bấm B2 để Pull Up trong mục Device và để FallingEdge trong mục Interrupt để thực hiện ngắt khi có sườn xuống.



Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	Pull Up	FallingEdge
Port_0_1	P0[1]	StdCPU	Pull Up	FallingEdge
Port_0_2	P0[2]	StdCPU	High Z Analog	DisableInt
Port_0_3	P0[3]	StdCPU	High Z Analog	FallingEdge
Port_0_4	P0[4]	StdCPU	High Z Analog	RisingEdge
Port_0_5	P0[5]	StdCPU	High Z Analog	ChangeFromRead
Port_0_6	P0[6]	StdCPU	High Z Analog	DisableInt
Port_1_3	P1[3]	StdCPU	High Z Analog	DisableInt
Port_1_4	P1[4]	StdCPU	Strong	DisableInt
Port_1_5	P1[5]	StdCPU	High Z Analog	DisableInt
Port_1_6	P1[6]	StdCPU	Strong	DisableInt
Port_1_7	P1[7]	StdCPU	High Z Analog	DisableInt
Port_2_0	P2[0]	StdCPU	Pull Up	DisableInt
Port_2_1	P2[1]	StdCPU	High Z Analog	DisableInt
Port_2_2	P2[2]	StdCPU	High Z Analog	DisableInt

Sau đó click Generate Application  và chuyển sang giao diện viết chương trình  :



Vào Source Files > boot.asm và tìm đến dòng:

```

org 1Ch    ;GPIO Interrupt Vector

ljmp PSoC_GPIO_ISR

reti
    
```

Ta sẽ thay dòng PsoC_GPIO_ISR bằng _PsoC_GPIO_ISR (có thêm dấu _ phía trước) hoặc thành tên bất kỳ mà ta muốn.

Rồi nhấn Ctrl + S.

Chú ý:

- Phân biệt chữ hoa và chữ thường.

Sau mỗi lần Generate Application lại thì phải làm lại bước trên.

3. Code chương trình.

// 16.6.2011

```

#include <m8c.h>    // part specific constants and macros
#include "PSoCAPI.h" // PSoC API definitions for all User Modules
// chương trình ngắt
#pragma interrupt_handler PSoC_GPIO_ISR
void PSoC_GPIO_ISR()
{
    if((PRT0DR & 0x01) == 0)// ngắt xảy ra trên chân P0.0
        PRT1DR = 0xef; // sang led D1 trên chân P1.4
}
    
```

```
    if((PRT0DR & 0x02) == 0)// ngat xay ra tren chan P0.1
        PRT1DR = 0xbf; // sang led D2 tren chan P1.6
    }

void main()
{
    M8C_EnableGInt; // cho phép ngắt
    M8C_EnableIntMask(INT_MSK0,0x20); // tắt ngắt
    PRT0DR = 0xff;
    PRT1DR = 0xff;
    PRT2DR = 0xff;
    while(1)
    {
        if((PRT2DR & 0x01) == 0)
            PRT1DR = 0xff; // tắt led
    }
}
```

V. Kết luận.

Kit thực hành đã được sử dụng trong đợt thực tập. Qua kết quả thử nghiệm có thể kết luận như sau :

- Về mặt thiết kế kỹ thuật, Kit đã đạt yêu cầu đặt ra, sử dụng tốt trong cả đợt thử nghiệm mà hầu như không có sai sót nào.
- Về hiệu năng sử dụng , Kit thực hành xây dựng đã phát huy ưu điểm khá lớn trong điều kiện không có điều kiện trang bị nhiều bộ thực hành. Hệ thống bài thực hành từ dễ đến khó phù hợp với nhiều trình độ đào tạo khác nhau.

VI. Tài liệu tham khảo.

Các tài liệu về hướng dẫn sử dụng, lập trình... Các project mẫu làm trên cơ sở PSoC và chỉ phục vụ PsoC

http://bmtbd.uct.edu.vn/library/tiki-list_file_gallery.php?galleryId=19