Báo cáo thực tập

Contents

[Lời nói đầu 2](#_Toc112745310)

[I.Tổng quan về kit thực hành. 3](#_Toc112745311)

[I.1.Hình ảnh kit thực hành: 3](#_Toc112745312)

[I.2.Kit thực hành gồm có: 3](#_Toc112745313)

[II.Phần mền biên dịch và nạp code PSoC Creater. 4](#_Toc112745314)

[III.Các bài thực hành. 4](#_Toc112745315)

[Bài 1. Led đơn. 4](#_Toc112745316)

[Bài 2. Led 7 thanh. 7](#_Toc112745317)

[Bài 3. Led ma trận. 13](#_Toc112745318)

[Bài 4. Biến trở. 21](#_Toc112745319)

[Bài 5. Đo nhiệt độ với LM35. 29](#_Toc112745320)

[Bài 6. Giao tiếp UART. 35](#_Toc112745321)

[Bài 7. Xác định thời gian với DS1307. 39](#_Toc112745322)

[Bài 8. Tạo xung PWM. 48](#_Toc112745323)

[Bài 9. Thiết lập Timer và ngắt Timer. 50](#_Toc112745324)

[Bài 10. Lcd. 52](#_Toc112745325)

[Bài 11. Nút nhấn. 55](#_Toc112745326)

[IV.Kết luận. 60](#_Toc112745327)

[V.Tài liệu tham khảo 60](#_Toc112745328)

# Lời nói đầu

Psoc hay Psoc Mixed-Signal Arrays là từ viết tắt của Programmable System-On-Chips. Psoc là chip mà có thể tích hợp cả vi điều khiển các thành phần tương tự và các thành phần số xung quanh vi điều khiển nhúng vào một hệ thống. Một chip đơn PSoC có thể tích hợp lên đến 100 chức năng ngoại vi với 1 vi điều khiển, làm giảm thời gian thiết kế, không gian board, năng lượng tiêu hao và giảm 5% giá thành sản phẩm ít nhất 10$ cho mỗi hệ thống.

Các chip chế tạo theo công nghệ Psoc cho phép thay đổi được cấu hình đơn giản bằng cách gán chức năng cho các khối tài nguyên có sẵn trên chip. Hơn nữa nó còn có thể kết nối tương đối mềm dẻo các khối chức năng với nhau hoặc giữa các khối chức năng với các cổng vào ra. Chính vì vậy mà Psoc có thể thay thế cho rất nhiều chức năng nền của một hệ thống cơ bản chỉ bằng duy nhất một chip đơn.

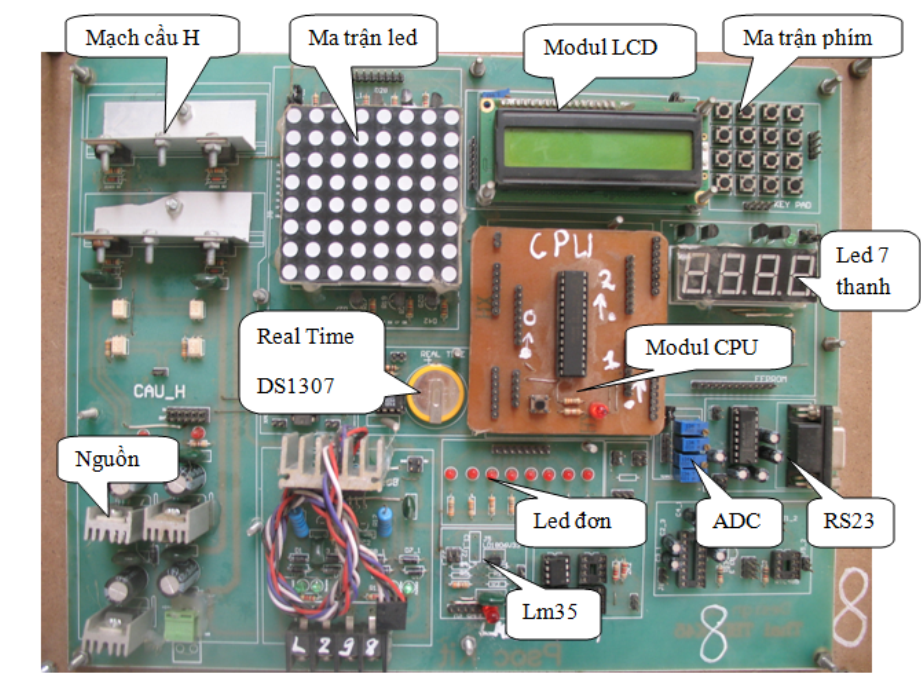
KIT phát triển PSoC đã được xây dựng cho phép người học có thể nhanh chóng xây dựng các ứng dụng trên cơ sở các dòng PSoC và các ngoại vi cơ bản. Báo cáo này thể hiện những kết quả đã làm được trong hai tuần đầu đi thực tập tại công ty Trang Bị Điện.

Mọi ý kiến đóng góp xin gửi về: Nguyễn Đức Đạt K59 - ĐHGTVT

Gmail: [nhiennhienbn@gmail.com](mailto:nhiennhienbn@gmail.com)

# I.Tổng quan về kit thực hành.

## I.1.Hình ảnh kit thực hành:



## I.2.Kit thực hành gồm có:

1.Module nguồn: Cung cấp uồn cho toàn bộ kit.

2.Module CPU

Module CPU gồm có:

- Chip PSoC 4 – CY8C4125AXI-483.

- Jump để nối mạch nạp để nạp chương trình từ máy tính xuống chip.

- Led báo nguồn khi module được cấp nguồn.

- Các Port của chip.

3.Module 8 led đơn.

4.Module 4 led 7 thanh.

5.Module ma trận phím.

6.Module led ma trận.

7.Module biến trở.

8.Module LM35.

9.Module giao tiếp UART

10.Module DS1307.

11.Module mạch cầu H.

12.Module EEPROM .

# II.Phần mền biên dịch và nạp code PSoC Creater.

PSoC Creator là IDE phần mềm thế hệ thứ hai để thiết kế gỡ lỗi và lập trình các thiết bị PSoC 3/4/5. IDE phát triển được kết hợp với một trình biên tập thiết kế đồ họa dễ sử dụng để tạo thành một môi trường đồng thiết kế phần cứng / phần mềm mạnh mẽ.

PSoC Creator bao gồm hai khối xây dựng cơ bản. Chương trình cho phép người dùng lựa chọn, cấu hình và kết nối các mạch hiện có trên chip và các thành phần tương đương với thiết bị ngoại vi trên MCU. Điều khiến PSoC trở nên hấp dẫn là khả năng tạo ra các thiết bị ngoại vi dành riêng cho ứng dụng trong phần cứng. Cypress xuất bản các gói thành phần nhiều lần trong năm.

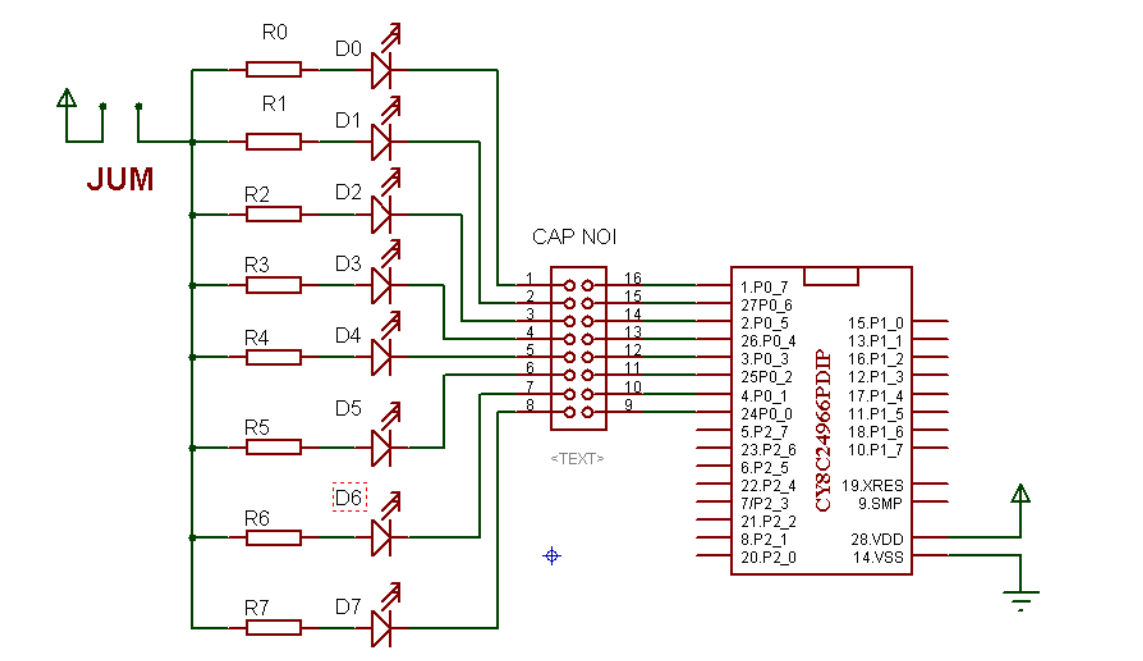
Người dùng PSoC nhận được thiết bị ngoại vi mới cho phần cứng hiện có của họ mà không bị tính phí hoặc phải mua phần cứng mới. PSoC Creator cũng cho phép nhiều tự do trong việc gán các thiết bị ngoại vi cho các chân I / O.

# III.Các bài thực hành.

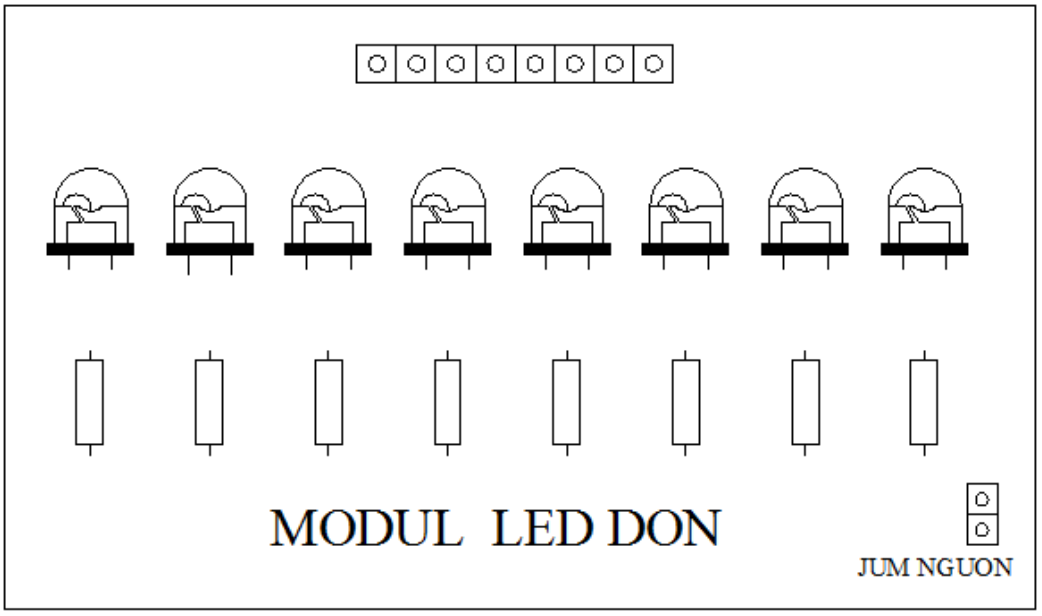
## Bài 1. Led đơn.

Mục tiêu là sử dụng Pin P1[0] của vi điều khiển để điều khiển 1 led đơn nhấp nháy.

1.Mạch nguyên lý.

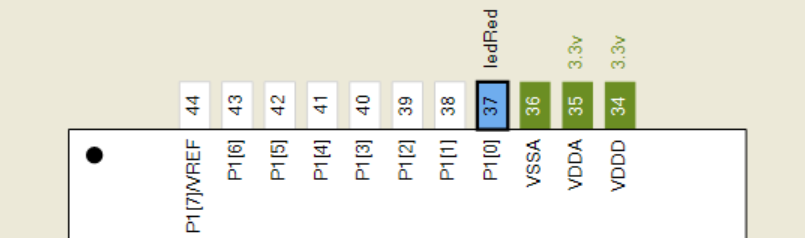


2.Mạch trên kit.

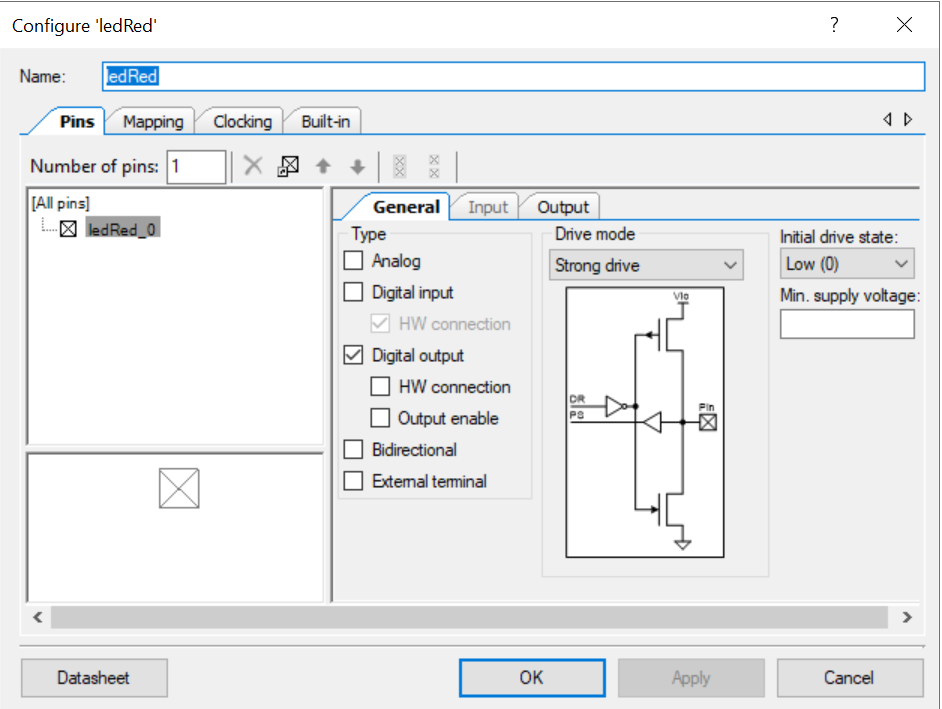


3.Kết nối cáp.

Nối Jump cấp nguồn cho module led, và nối cáp giữa pin P1[0] của vi điều khiển với led.



4.Viết chương trình.



File ledSingle.h

#ifndef LEDSINGLE\_H

#define LEDSINGLE\_H

#include "project.h"

void ledBlink();

void ledTurnOn();

void ledTurnOff();

#endif

File ledSingle.c

#include "ledSingle.h"

void ledBlink()

{

ledRed\_Write( ~ledRed\_Read());

CyDelay(300);

}

void ledTurnOn()

{

ledRed\_Write(0);

}

void ledTurnOff()

{

ledRed\_Write(1);

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "project.h"

#include "ledSingle.h"

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

for(;;)

{

ledTurnOn();

CyDelay(1000);

ledTurnOff();

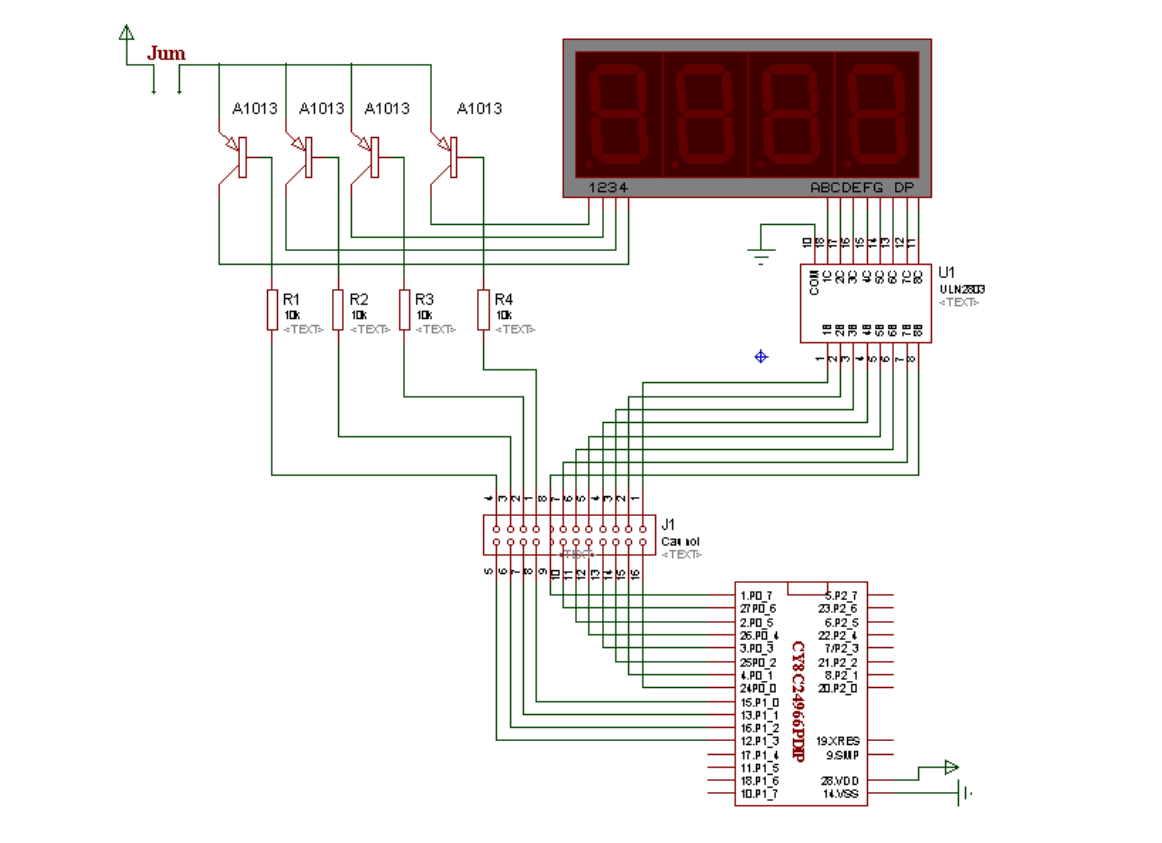
CyDelay(1000);

}

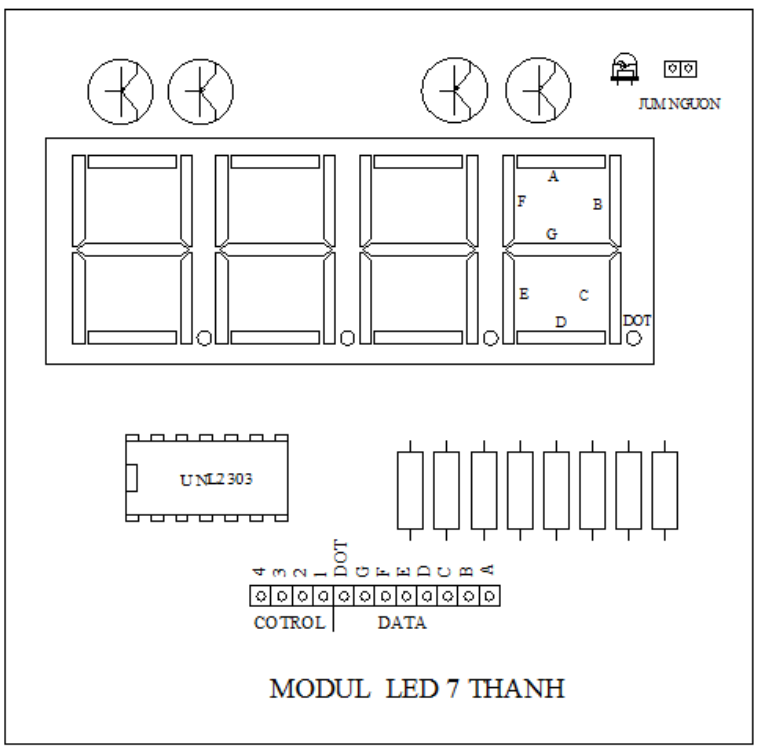
}

## Bài 2. Led 7 thanh.

1.Mạch nguyên lý.



2.Mạch trên kit.



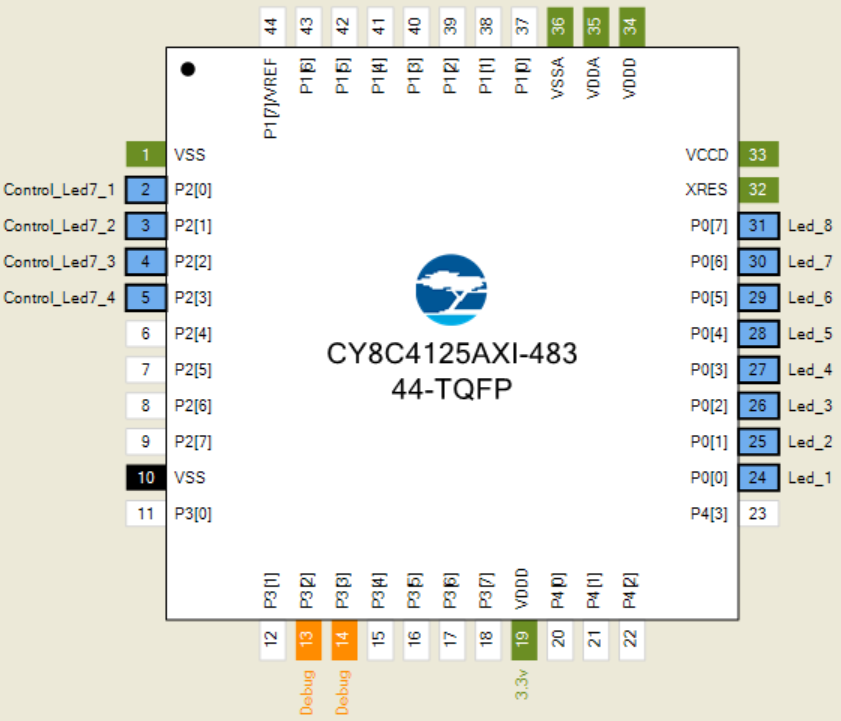
3.Kết nối cáp.

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 P2.0 nối với chân control của led thứ nhất ..... P2.3 nối với chân control của led thứ nhất 4.



4.Viết chương trình.

File Led7\_Seg.h

#ifndef LED7\_SEG\_H

#define LED7\_SEG\_H

#include <stdio.h>

#include "project.h"

enum Status\_pin

{

low = 0,

hight

};

void scanLed();

void check\_Led\_Live\_Or\_Die(uint8 location, uint8 data);

#endif

File Led7\_Seg.c

#include "Led7\_Seg.h"

uint8 writeHightOrLow(uint8 dataNeedWrite, uint8 locationWrite);

void write\_data(uint8 dataNeedWrite);

void control\_Led\_Light(int location);

void turnOffAllLed();

unsigned char code\_Led7\_Seg[] = {0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6};

uint8 writeHightOrLow(uint8 dataNeedWrite, uint8 locationWrite)

{

uint8 result = 0;

switch(locationWrite)

{

case 0:

((dataNeedWrite & (1<<0)) == (1<<0) ) ? (result = hight) : (result = low);

break;

case 1:

((dataNeedWrite & (1<<1)) == (1<<1)) ? (result = hight) : (result = low);

break;

case 2:

((dataNeedWrite & (1<<2)) == (1<<2)) ? (result = hight) : (result = low);

break;

case 3:

((dataNeedWrite & (1<<3)) == (1<<3) ) ? (result = hight) : (result = low);

break;

case 4:

((dataNeedWrite & (1<<4)) == (1<<4) ) ? (result = hight) : (result = low);

break;

case 5:

((dataNeedWrite & (1<<5)) == (1<<5) ) ? (result = hight) : (result = low);

break;

case 6:

((dataNeedWrite & (1<<6)) == (1<<6) ) ? (result = hight) : (result = low);

break;

case 7:

((dataNeedWrite & (1<<7)) == (1<<7) ) ? (result = hight) : (result = low);

}

return result;

}

void write\_data(uint8 dataNeedWrite)

{

Led\_1\_Write( writeHightOrLow(dataNeedWrite, 0) );

Led\_2\_Write( writeHightOrLow(dataNeedWrite, 1) );

Led\_3\_Write( writeHightOrLow(dataNeedWrite, 2) );

Led\_4\_Write( writeHightOrLow(dataNeedWrite, 3) );

Led\_5\_Write( writeHightOrLow(dataNeedWrite, 4) );

Led\_6\_Write( writeHightOrLow(dataNeedWrite, 5) );

Led\_7\_Write( writeHightOrLow(dataNeedWrite, 6) );

Led\_8\_Write( writeHightOrLow(dataNeedWrite, 7) );

}

void control\_Led\_Light(int location)

{

switch(location)

{

case 1:

Control\_Led7\_1\_Write(0);

Control\_Led7\_2\_Write(1);

Control\_Led7\_3\_Write(1);

Control\_Led7\_4\_Write(1);

break;

case 2:

Control\_Led7\_1\_Write(1);

Control\_Led7\_2\_Write(0);

Control\_Led7\_3\_Write(1);

Control\_Led7\_4\_Write(1);

break;

case 3:

Control\_Led7\_1\_Write(1);

Control\_Led7\_2\_Write(1);

Control\_Led7\_3\_Write(0);

Control\_Led7\_4\_Write(1);

break;

case 4:

Control\_Led7\_1\_Write(1);

Control\_Led7\_2\_Write(1);

Control\_Led7\_3\_Write(1);

Control\_Led7\_4\_Write(0);

break;

}

}

void turnOffAllLed()

{

Control\_Led7\_1\_Write(1);

Control\_Led7\_2\_Write(1);

Control\_Led7\_3\_Write(1);

Control\_Led7\_4\_Write(1);

}

void scanLed(unsigned int number)

{

uint8 nghin = number/1000;

uint8 tram = (number%1000)/100;

uint8 chuc = (number%100)/10;

uint8 donvi = (number%10);

// Display led 1

control\_Led\_Light(4);

write\_data(code\_Led7\_Seg[nghin]);

CyDelay(5);

turnOffAllLed();

// Display led 2

control\_Led\_Light(3);

write\_data(code\_Led7\_Seg[tram]);

CyDelay(5);

turnOffAllLed();

// Display led 3

control\_Led\_Light(2);

write\_data(code\_Led7\_Seg[chuc]);

CyDelay(5);

turnOffAllLed();

// Display led 4

control\_Led\_Light(1);

write\_data(code\_Led7\_Seg[donvi]);

CyDelay(5);

turnOffAllLed();

}

void check\_Led\_Live\_Or\_Die(uint8 location, uint8 data)

{

control\_Led\_Light(location);

write\_data(data);

CyDelay(5);

turnOffAllLed();

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include "project.h"

#include "Led7\_Seg.h"

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

for(;;)

{

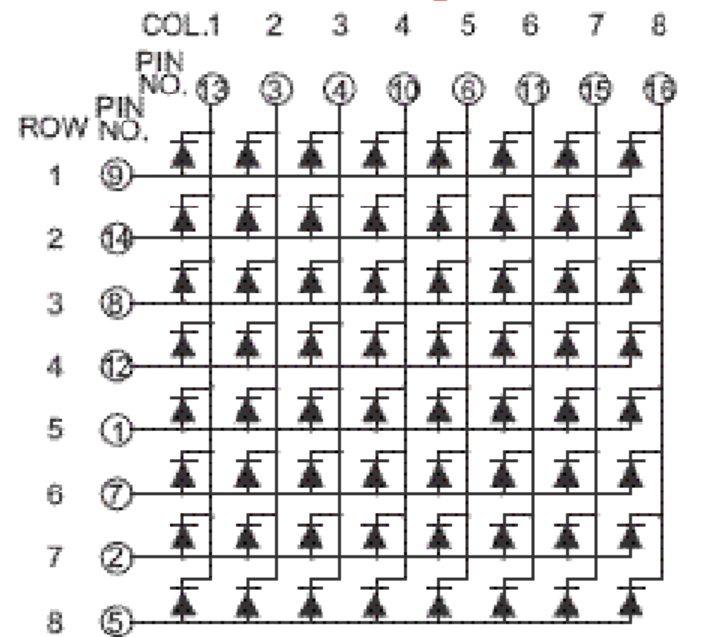
scanLed(7896);

}

}

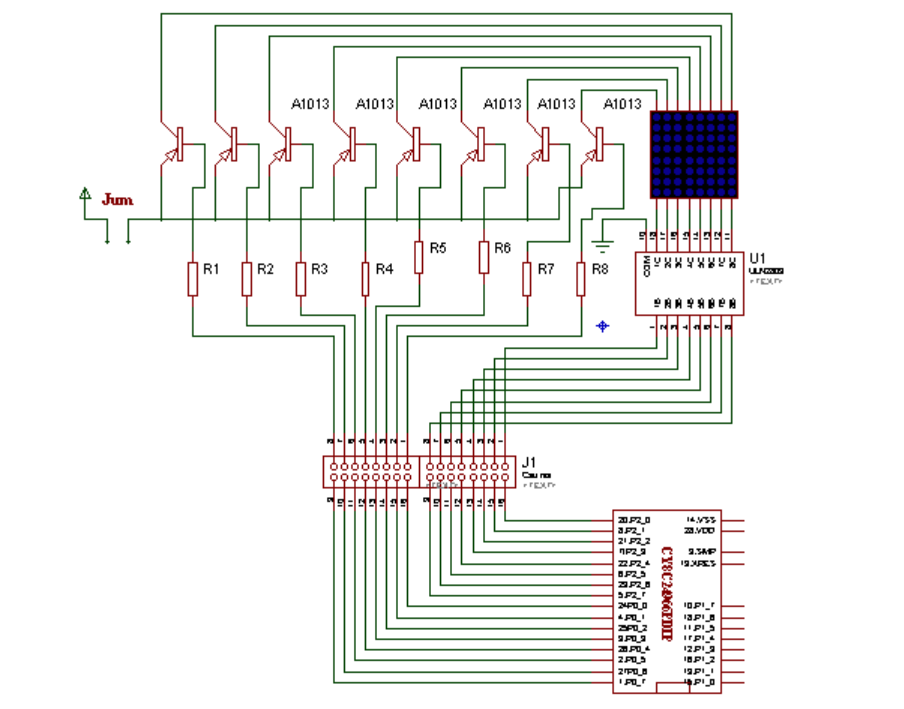
## Bài 3. Led ma trận.

0.Cấu tạo led ma trận

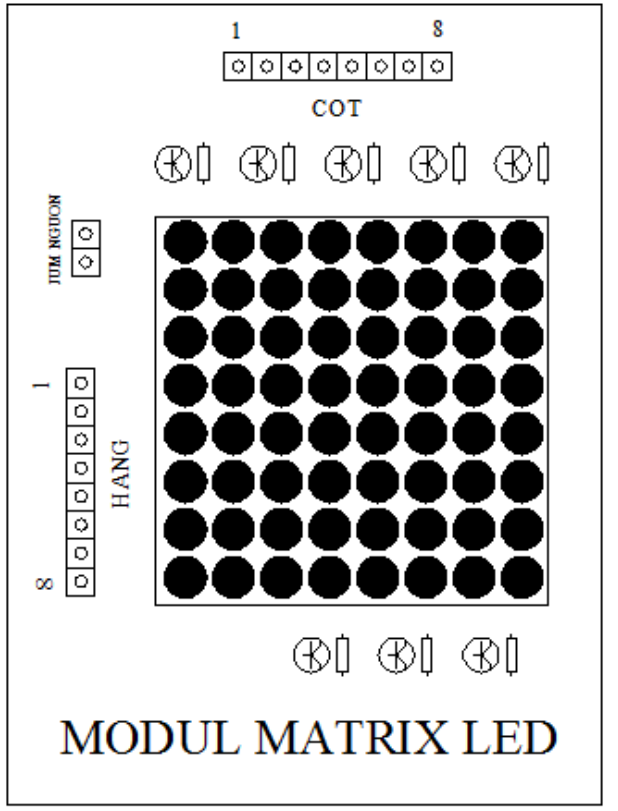


Ma trận Led 8x8. Các Led được nối với nhau thành 8 hàng và 8 cột. Muốn cho 1 led sáng thì ta phải đưa hàng của nó lên 1 và cột tương ứng xuống 0.

1.Mạch nguyên lý.



2.Mạch trên kit.

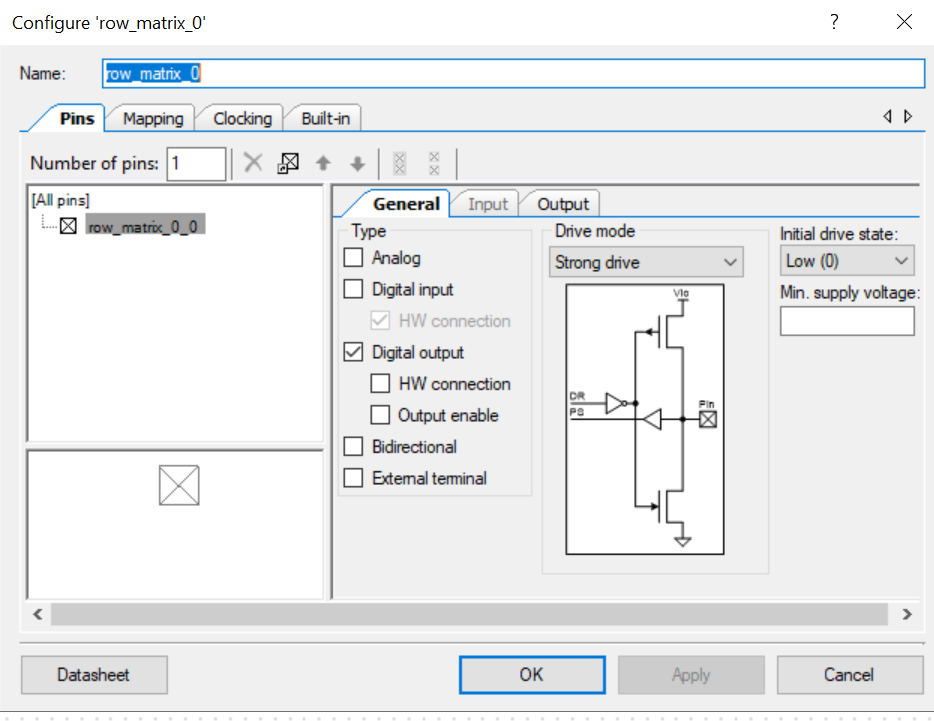


3.Kết nối cáp.

Nối Port1 của VDK với các hàng của ma trận led.

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





4.Viết chương trình.

File ledMatrix.h

#ifndef MATRIX\_H

#define MATRIX\_H

#include "project.h"

enum row\_or\_column

{

row=0,

column

};

enum Status\_pin

{

low = 0,

hight

};

void light\_character\_k(int k);

void light\_from\_0\_to\_9();

#endif

File ledMatrix.c

#include "ledMatrix.h"

uint8 control[] = { 0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80 };

uint8 data[] = {

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

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

0xff,0x86,0xb6,0xb6,0xb6,0xb6,0xb0,0xff,//2

0xff,0x66,0x66,0x66,0x66,0x98,0xff,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

0xe7,0xe7,0xe7,0x00,0x00,0xe7,0xe7,0xe7,//10 --- +

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,//11-turn on full

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//12-turn off full

0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//13-

0xFF,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//14-

0xFF,0xFF,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,//15-

0xFF,0xFF,0xFF,0x00,0xFF,0xFF,0xFF,0xFF,//16-

0xFF,0xFF,0xFF,0xFF,0x00,0xFF,0xFF,0xFF,//17-

0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0xFF,0xFF,//18-

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0xFF,//19

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,//20-

0x7f,0x7f,0x7f,0x7f,0x7f,0x7f,0x7f,0x7f //21

};

/\*

Current location --> will return hight logic, contra return low logic

\*/

uint8 writeHightOrLow(uint8 dataNeedWrite, uint8 locationWrite)

{

uint8 result = 0;

switch(locationWrite)

{

case 0:

((dataNeedWrite & (1<<0)) == (1<<0) ) ? (result = hight) : (result = low);

break;

case 1:

((dataNeedWrite & (1<<1)) == (1<<1)) ? (result = hight) : (result = low);

break;

case 2:

((dataNeedWrite & (1<<2)) == (1<<2)) ? (result = hight) : (result = low);

break;

case 3:

((dataNeedWrite & (1<<3)) == (1<<3) ) ? (result = hight) : (result = low);

break;

case 4:

((dataNeedWrite & (1<<4)) == (1<<4) ) ? (result = hight) : (result = low);

break;

case 5:

((dataNeedWrite & (1<<5)) == (1<<5) ) ? (result = hight) : (result = low);

break;

case 6:

((dataNeedWrite & (1<<6)) == (1<<6) ) ? (result = hight) : (result = low);

break;

case 7:

((dataNeedWrite & (1<<7)) == (1<<7) ) ? (result = hight) : (result = low);

}

return result;

}

void write\_data(uint8 dataNeedWrite, uint8 rowOrCloumn)

{

switch(rowOrCloumn)

{

case row:

row\_matrix\_0\_Write( writeHightOrLow(dataNeedWrite, 0) );

row\_matrix\_1\_Write( writeHightOrLow(dataNeedWrite, 1) );

row\_matrix\_2\_Write( writeHightOrLow(dataNeedWrite, 2) );

row\_matrix\_3\_Write( writeHightOrLow(dataNeedWrite, 3) );

row\_matrix\_4\_Write( writeHightOrLow(dataNeedWrite, 4) );

row\_matrix\_5\_Write( writeHightOrLow(dataNeedWrite, 5) );

row\_matrix\_6\_Write( writeHightOrLow(dataNeedWrite, 6) );

row\_matrix\_7\_Write( writeHightOrLow(dataNeedWrite, 7) );

break;

case column:

col\_matrix\_0\_Write( writeHightOrLow(dataNeedWrite, 0) );

col\_matrix\_1\_Write( writeHightOrLow(dataNeedWrite, 1) );

col\_matrix\_2\_Write( writeHightOrLow(dataNeedWrite, 2) );

col\_matrix\_3\_Write( writeHightOrLow(dataNeedWrite, 3) );

col\_matrix\_4\_Write( writeHightOrLow(dataNeedWrite, 4) );

col\_matrix\_5\_Write( writeHightOrLow(dataNeedWrite, 5) );

col\_matrix\_6\_Write( writeHightOrLow(dataNeedWrite, 6) );

col\_matrix\_7\_Write( writeHightOrLow(dataNeedWrite, 7) );

break;

}

}

void light\_character\_k(int k)

{

int n;

write\_data(0xFF, column);

write\_data(0x00, row);

for(n=0;n<8;n++)

{

write\_data( data[k\*8 + n], column );

write\_data( control[n], row );

CyDelay(2);

}

}

void light\_from\_0\_to\_9()

{

int run , loops;

write\_data(0xFF, column);

write\_data(0x00, row);

while(1)

{

for(run=0; run<10; run++)

{

for(loops=0; loops<500; loops++)

light\_character\_k(run);

write\_data(0xFF, column);

write\_data(0x00, row);

}

}

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "project.h"

#include "ledMatrix.h"

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

for(;;)

{

light\_from\_0\_to\_9();

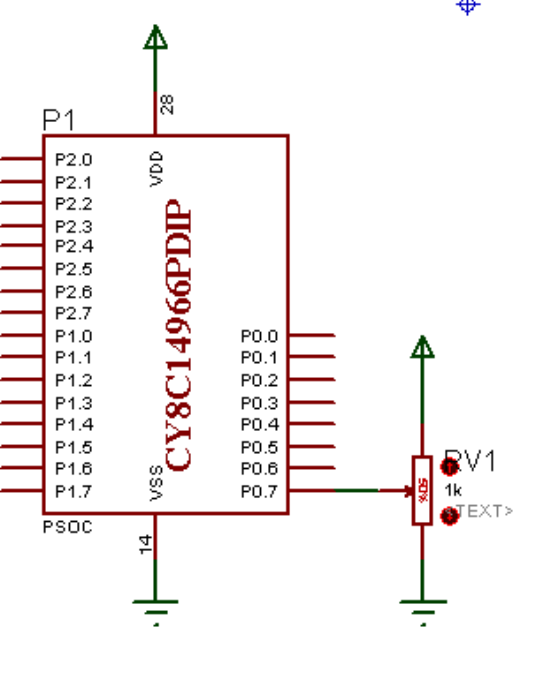
//light\_character\_k(11);

}

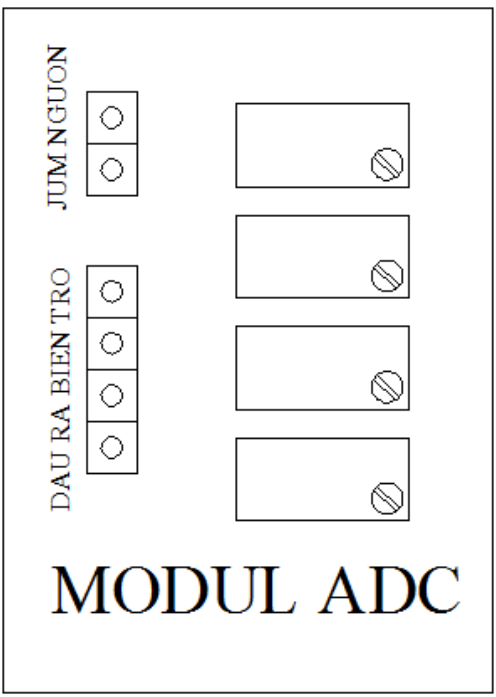
}

## Bài 4. Biến trở.

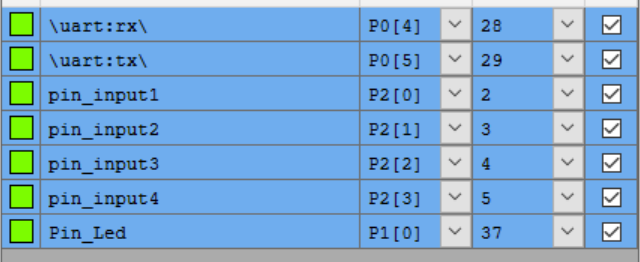
1.Mạch nguyên lý.



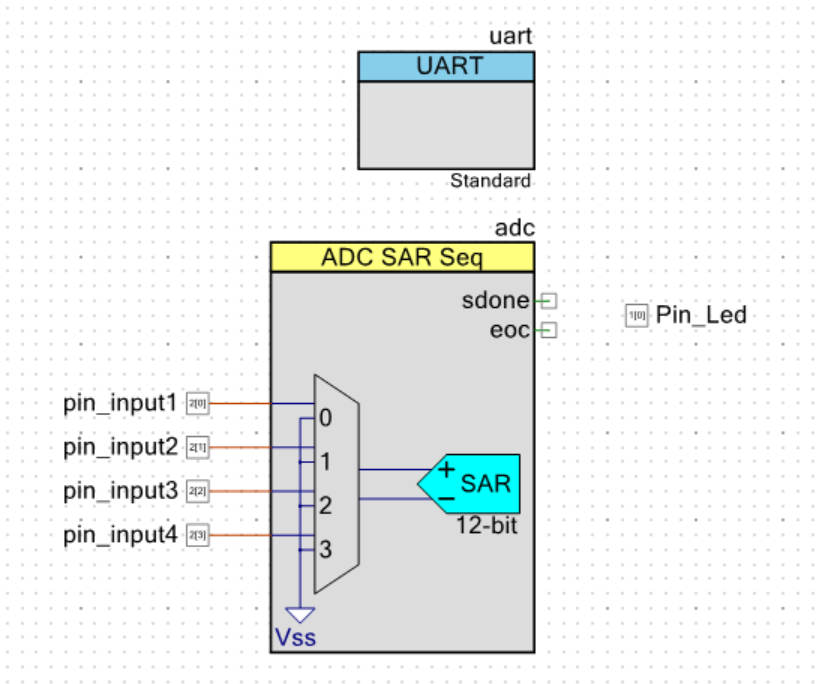
2.Mạch trên kit.

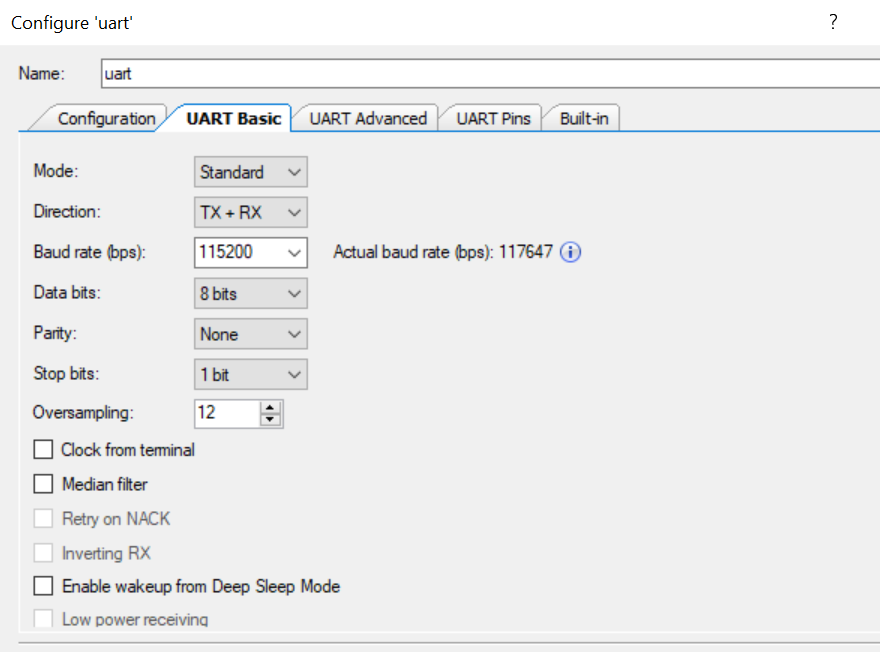


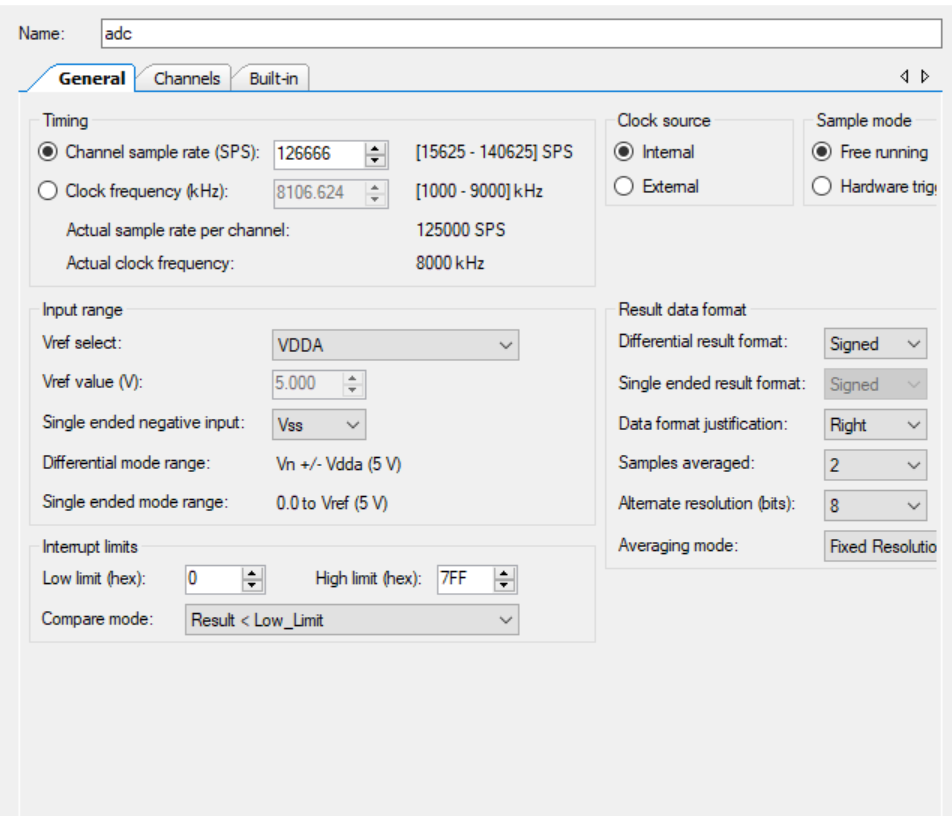
3.Kết nối cáp.

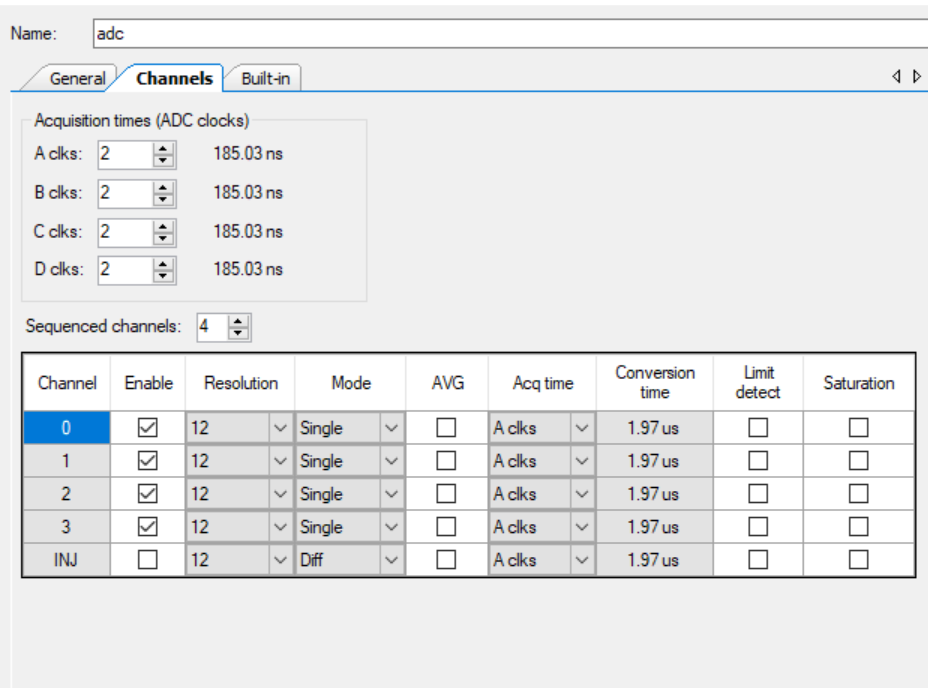


4.Viết chương trình.









File measureResister.h

#ifndef MEASURERESISTER\_H

#define MEASURERESISTER\_H

#include <stdio.h>

#include "project.h"

float read\_adc(int channel);

#endif

File measureResister.c

#include "measureResister.h"

float read\_adc(int channel)

{

float value\_read\_adc = 0;

switch(channel)

{

case 0:

adc\_StartConvert();

adc\_IsEndConversion(adc\_WAIT\_FOR\_RESULT);

value\_read\_adc = adc\_GetResult16(0);

adc\_StopConvert();

break;

case 1:

adc\_StartConvert();

adc\_IsEndConversion(adc\_WAIT\_FOR\_RESULT);

value\_read\_adc = adc\_GetResult16(1);

adc\_StopConvert();

break;

case 2:

adc\_StartConvert();

adc\_IsEndConversion(adc\_WAIT\_FOR\_RESULT);

value\_read\_adc = adc\_GetResult16(2);

adc\_StopConvert();

break;

case 3:

adc\_StartConvert();

adc\_IsEndConversion(adc\_WAIT\_FOR\_RESULT);

value\_read\_adc = adc\_GetResult16(3);

adc\_StopConvert();

}

// Choose --> data format justification: right

float value\_Adc\_mVolt = adc\_CountsTo\_mVolts(0, value\_read\_adc);

//float value\_Adc\_mVolt = (value\_read\_adc/127)\*5\*1000;

return value\_Adc\_mVolt;

}

File uart\_transmit.h

#ifndef UART\_TRANSMIT\_H

#define UART\_TRANSMIT\_H

#include <stdio.h>

#include "project.h"

void test\_transmit\_data();

void transmit\_data\_to\_computer(float data\_need\_convert,int location);

#endif

File uart\_transmit.c

#include "uart\_transmit.h"

void test\_transmit\_data()

{

char data\_test[] = "\nTransmit OK\n";

uart\_UartPutString(data\_test);

}

void transmit\_data\_to\_computer(float data\_need\_convert,int location)

{

char string\_result[40];

switch(location)

{

case 0:

sprintf(string\_result, "Value1:%0.3f\n", data\_need\_convert);

break;

case 1:

sprintf(string\_result, "Value2:%0.3f\n", data\_need\_convert);

break;

case 2:

sprintf(string\_result, "Value3:%0.3f\n", data\_need\_convert);

break;

case 3:

sprintf(string\_result, "Value4:%0.3f\n", data\_need\_convert);

}

uart\_UartPutString(string\_result);

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include "project.h"

#include "measureResister.h"

#include "uart\_transmit.h"

void init\_system();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

init\_system();

for(;;)

{

test\_transmit\_data();

transmit\_data\_to\_computer(read\_adc(0), 0);

transmit\_data\_to\_computer(read\_adc(1), 1);

transmit\_data\_to\_computer(read\_adc(2), 2);

transmit\_data\_to\_computer(read\_adc(3), 3);

CyDelay(1000);

}

}

void init\_system()

{

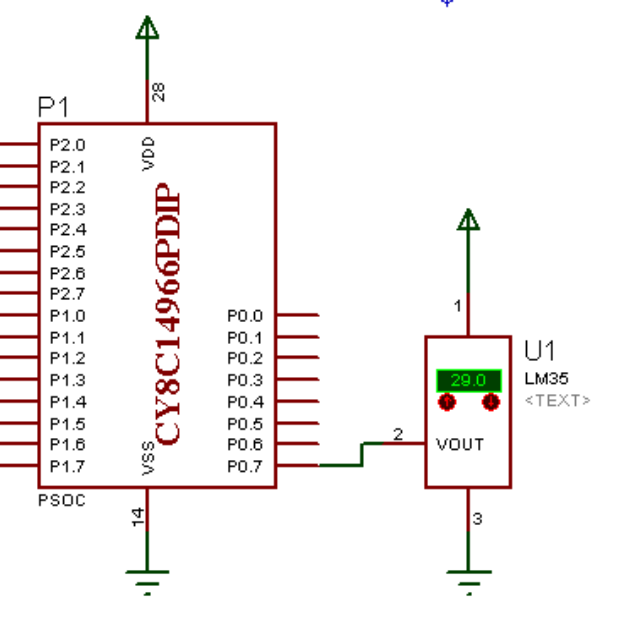
uart\_Start();

adc\_Start();

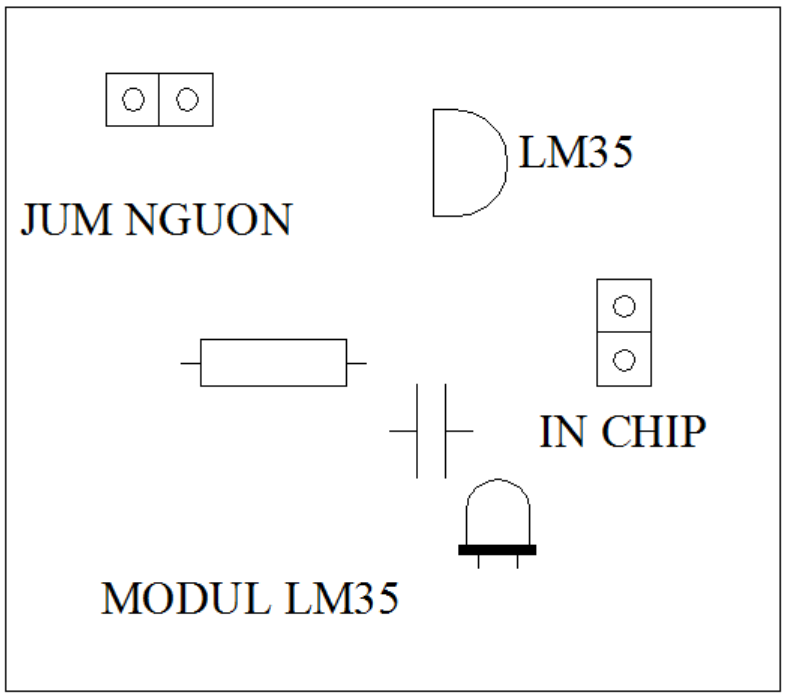
}

## Bài 5. Đo nhiệt độ với LM35.

1.Mạch nguyên lý.

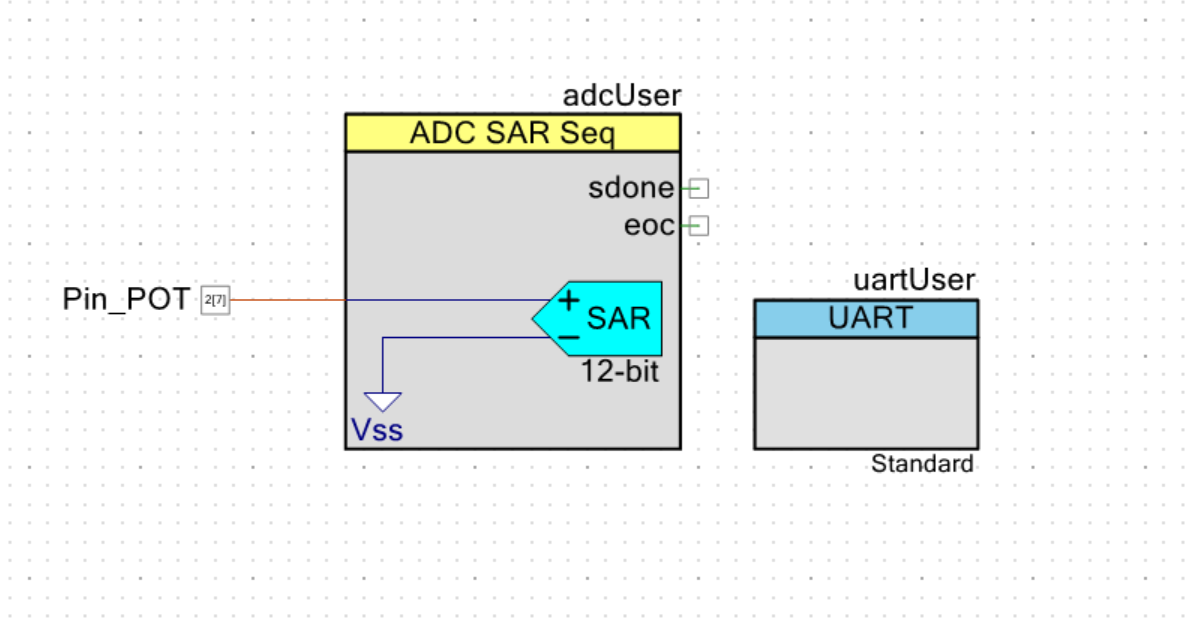


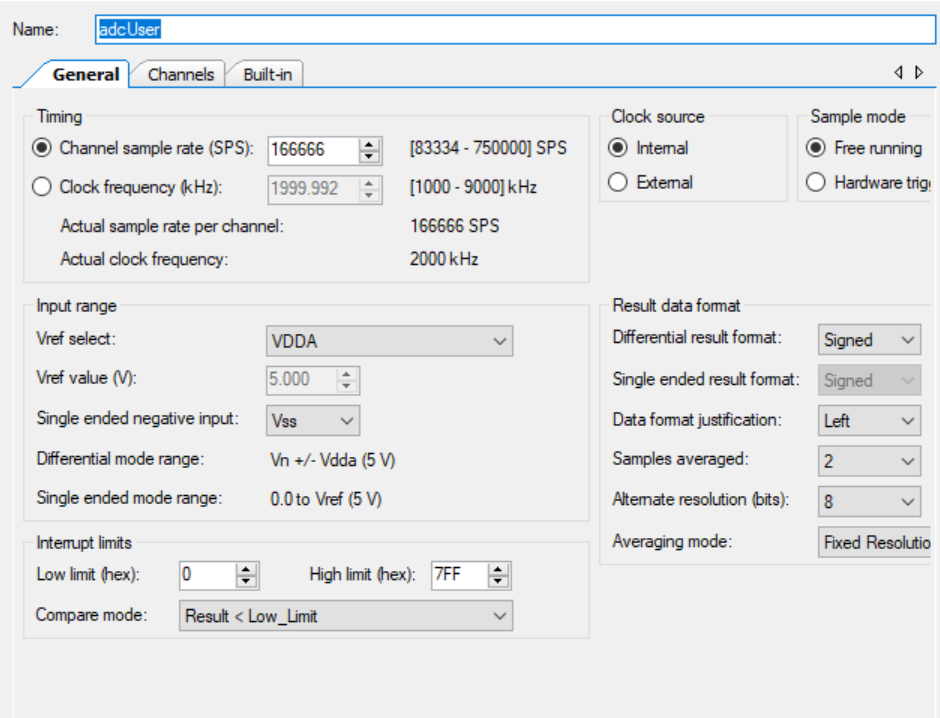
2.Mạch trên kit.

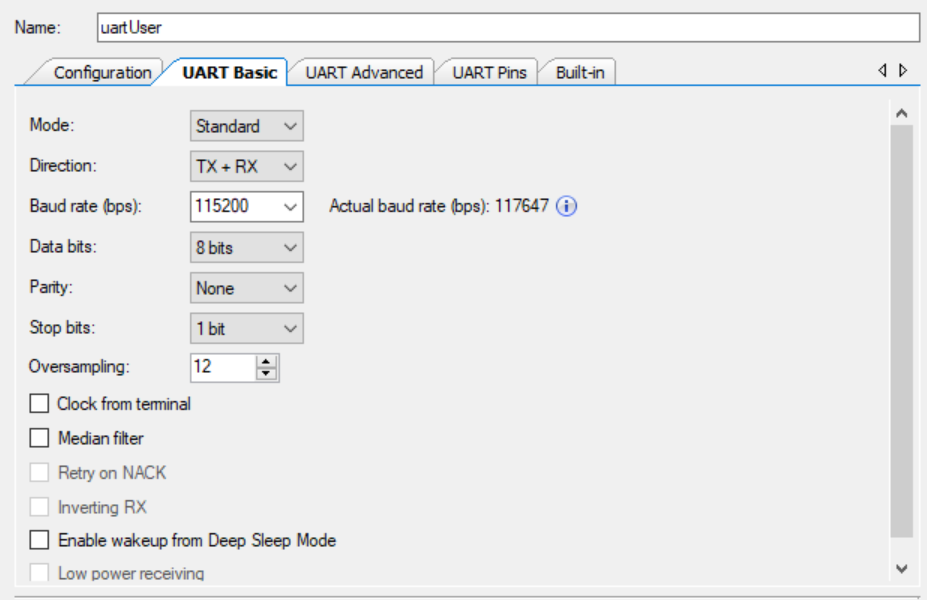


3.Kết nối cáp.

4.Viết chương trình.







File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include "project.h"

void Init\_System();

float readADC();

void transmitToPC(float adcData);

float readLM35();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

Init\_System();

float result\_Read\_Adc;

for(;;)

{

//result\_Read\_Adc = readADC();

result\_Read\_Adc = readLM35();

transmitToPC( result\_Read\_Adc );

CyDelay( 1000 );

}

}

void Init\_System()

{

uartUser\_Start();

adcUser\_Start(); /\* Initialize ADC \*/

}

float readADC()

{

float value\_read\_adc=0;

adcUser\_StartConvert(); /\* Start ADC conversions \*/

adcUser\_IsEndConversion( adcUser\_WAIT\_FOR\_RESULT );

value\_read\_adc = adcUser\_GetResult16(0);

adcUser\_StopConvert();

// Choose --> data format justification: right

//float value\_Adc\_mVolt = adcUser\_CountsTo\_mVolts(0, value\_read\_adc);

//float value\_Adc\_mVolt = (value\_Adc\_Volt/127)\*5\*1000;

// Choose --> data format justification: left

float value\_Adc\_mVolt = (value\_read\_adc/32767)\*5\*1000;

return value\_Adc\_mVolt;

}

void transmitToPC(float adcData)

{

char buffer\_value\_transmit[20];

sprintf(buffer\_value\_transmit,"%f\n", adcData);

uartUser\_UartPutString(buffer\_value\_transmit);

}

float readLM35()

{

float value\_read\_adc=0;

adcUser\_StartConvert(); /\* Start ADC conversions \*/

adcUser\_IsEndConversion( adcUser\_WAIT\_FOR\_RESULT );

value\_read\_adc = adcUser\_GetResult16(0);

adcUser\_StopConvert();

// Choose --> data format justification: right

//float value\_Adc\_mVolt = adcUser\_CountsTo\_mVolts(0, value\_read\_adc);

//float value\_Adc\_mVolt = (value\_Adc\_Volt/127)\*5\*1000;

// Choose --> data format justification: left

float value\_Adc\_mVolt = (value\_read\_adc/32767)\*5\*1000;

//1\*c == 10mv

// --> temprature = value\_Adc\_mVolt/10;

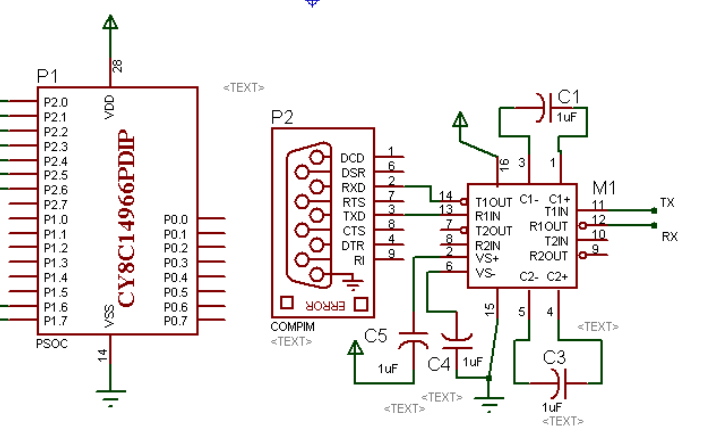
value\_Adc\_mVolt /=10;

return value\_Adc\_mVolt;

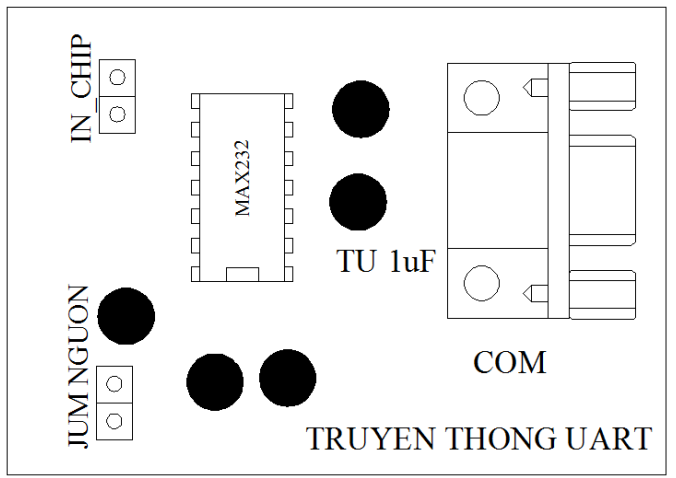
}

## Bài 6. Giao tiếp UART.

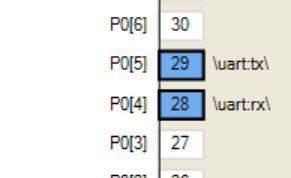
1.Mạch nguyên lý.



2.Mạch trên kit.



3.Kết nối cáp.



4.Viết chương trình.

File applicationUartt.h

#ifndef APPLICATIONUART\_H

#define APPLICATIONUART\_H

#include <stdio.h>

#include "project.h"

void printNumber(int number);

void printNumberIncrement();

void receive\_After\_Transmit\_Char\_Type();

void receive\_After\_Transmit\_String();

#endi

File applicationUartt.c

#include "applicationUart.h"

void printNumber(int number)

{

char strings[30];

sprintf(strings, "%d\n", number);

uart\_UartPutString(strings);

}

void printNumberIncrement()

{

char strings[30];

static int number = 10;

number++;

if(number > 15)

number = 5;

sprintf(strings, "%d\n", number);

uart\_UartPutString(strings);

}

void receive\_After\_Transmit\_Char\_Type()

{

static int i=0;

char txData[] = "\nPlear enter a character: ";

if( i==0 )

uart\_UartPutString( txData );

uint32 rxData;

rxData = uart\_UartGetChar(); // store received characters in temporary variable

if( rxData ) // make sure data is non-zero

{

uart\_UartPutString("\nResult: ");

uart\_UartPutChar( rxData ); // echo characters in terminal window

i=0;

return;

}

i=1;

// Handle received characters

}

/\*

void receive\_After\_Transmit\_String()

{

static int i=0;

int j;

char txData[] = "\nPlear enter a string: ";

char txDataString[20];

uint32 txDataReceive;

if(i==0)

uart\_UartPutString( txData );

txDataReceive = uart\_UartGetChar();

if(txDataReceive)

{

sprintf(txDataString, "%lu", txDataReceive);

uart\_UartPutString( uart\_UartGetChar() );

}

}

\*/

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "project.h"

#include "applicationUart.h"

void uartHandle();

void configSystem();

void uartHandle();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

configSystem();

for(;;)

{

receive\_After\_Transmit\_Char\_Type();

// receive\_After\_Transmit\_String();

CyDelay(1000);

}

}

void configSystem()

{

uart\_Start();

}

void uartHandle()

{

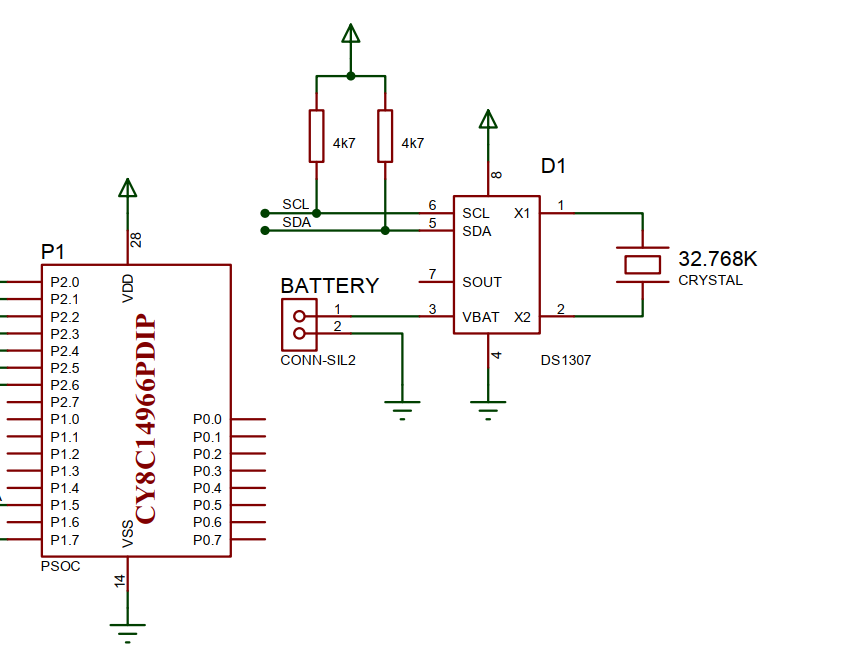
printNumberIncrement();

CyDelay(1000);

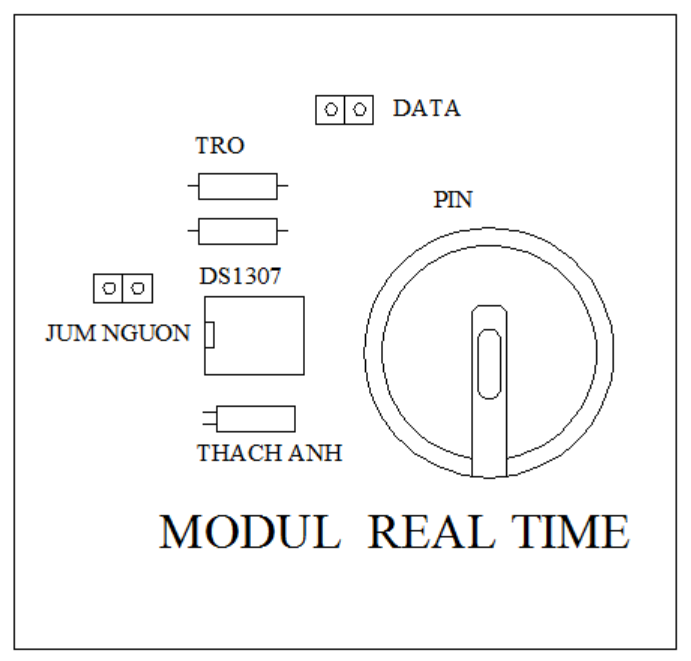
}

## Bài 7. Xác định thời gian với DS1307.

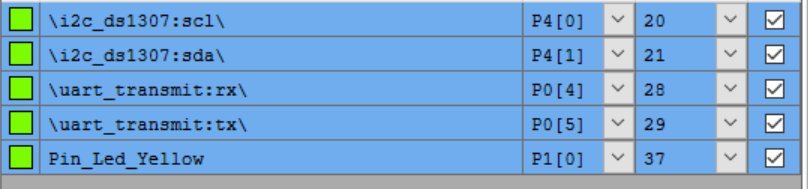
1.Mạch nguyên lý.



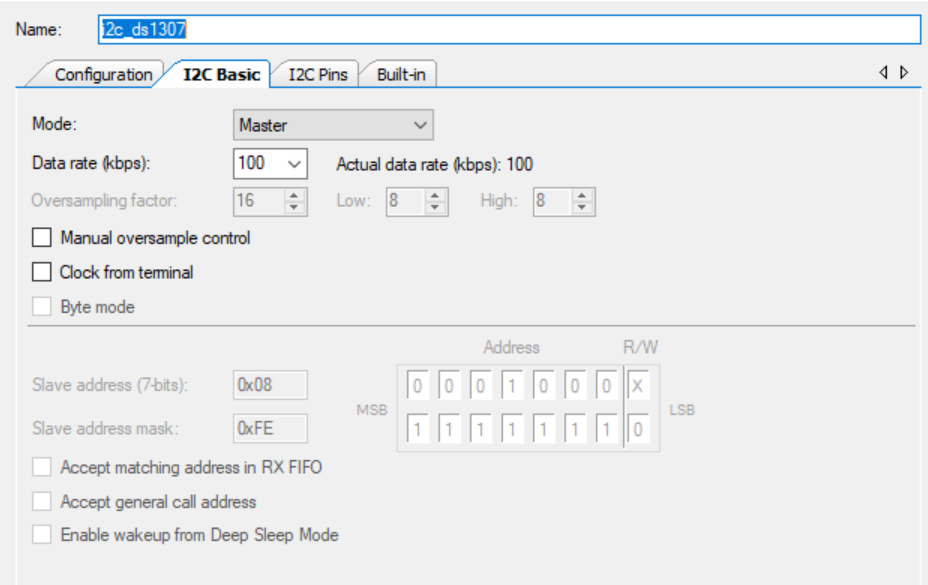
2.Mạch trên kit.

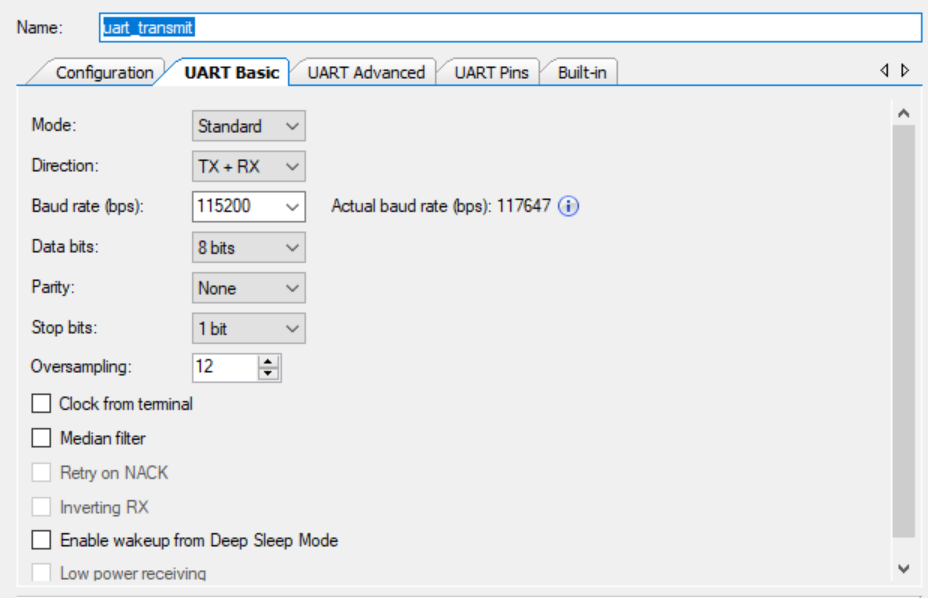


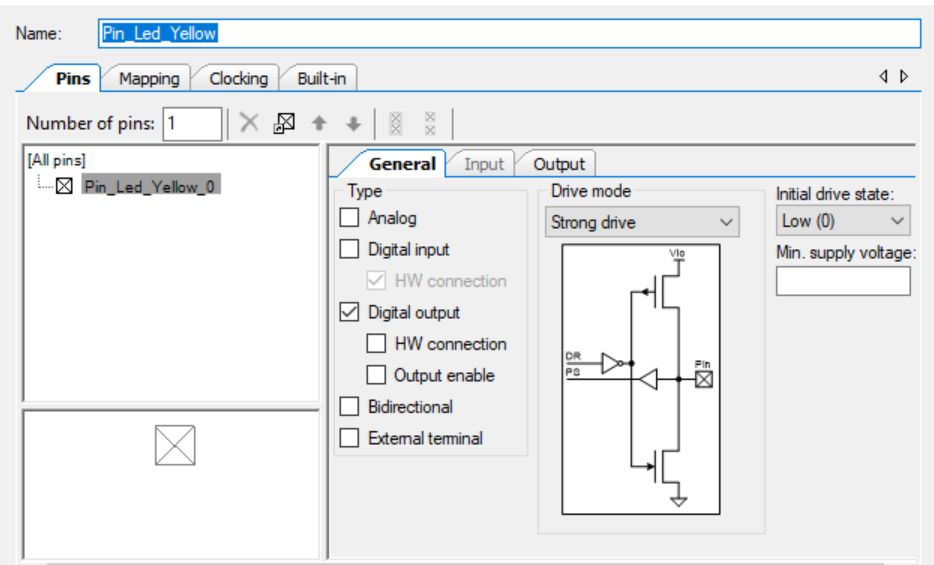
3.Kết nối cáp.



4.Viết chương trình.







File ds1307\_i2c.h

#ifndef DS1307\_I2C\_H

#define DS1307\_I2C\_H

#include "project.h"

typedef struct DataTime

{

uint8 second, minute, hour, \

day, date, month, year;

}data\_time;

enum DayOfWeek

{

Sun = 1,

Mon,

Tue,

Web,

Thur,

Fri,

Sat

};

#define ADDRESS\_SLAVE\_DS1307 0x68

void ds1307\_time\_init(data\_time \*time);

void ds1307\_read\_data(data\_time \*time);

void ds1307\_write\_data(data\_time \*time);

#endif

File ds1307\_i2c.c

#include "ds1307\_i2c.h"

uint8 BCD\_to\_DEC(uint8 data);

uint8 DEC\_to\_BCD(uint8 data);

uint8 BCD\_to\_DEC(uint8 data)

{

return (data>>4)\*10 + (data&0x0f);

}

uint8 DEC\_to\_BCD(uint8 data)

{

return (data/10)<<4 | (data%10);

}

/\*

- Dựa vào bảng Timekeeper registers, chúng ta sẽ thấy thanh ghi second có bit 7 CH (dùng để tạm dừng dao động),

chúng ta sẽ không đọc bit này nên sẽ cần biểu thức ( temp[0] & 0x7F ) để loại đi bit 7 này.

- Thanh ghi hour có bit 6 để chọn chế độ 12/24h, nếu bit 6 = 1, chế độ 12h sẽ được chọn.

Chúng ta sẽ không đọc bit 6 nên cần biểu thức temp[0] & 0x3F để loại bit 6 này.

\*/

void ds1307\_read\_data(data\_time \*time)

{

uint8 result, i;

uint8 temp[8];

do{

result = i2c\_ds1307\_I2CMasterSendStart(ADDRESS\_SLAVE\_DS1307, i2c\_ds1307\_I2C\_WRITE\_XFER\_MODE, 1);

}while(result != i2c\_ds1307\_I2C\_MSTR\_NO\_ERROR);

i2c\_ds1307\_I2CMasterWriteByte(0x00, 1);

i2c\_ds1307\_I2CMasterSendRestart(ADDRESS\_SLAVE\_DS1307, i2c\_ds1307\_I2C\_READ\_XFER\_MODE, 1);

for(i=0; i<6; i++)

i2c\_ds1307\_I2CMasterReadByte( i2c\_ds1307\_I2C\_ACK\_DATA, &temp[i] , 1);

i2c\_ds1307\_I2CMasterReadByte( i2c\_ds1307\_I2C\_NAK\_DATA, &temp[i] , 1);

i2c\_ds1307\_I2CMasterSendStop(1);

// Convert BCD to DEC

time->second = BCD\_to\_DEC(temp[0] & 0x7F);

time->minute = BCD\_to\_DEC(temp[1]);

time->hour = BCD\_to\_DEC(temp[2] & 0x3F); // 24h mode

time->day = BCD\_to\_DEC(temp[3]);

time->date = BCD\_to\_DEC(temp[4]);

time->month = BCD\_to\_DEC(temp[5]);

time->year = BCD\_to\_DEC(temp[6]);

}

void ds1307\_write\_data(data\_time \*time)

{

uint8 result, i;

uint32 temp[8];

// Convert DEC to BCD

temp[0] = DEC\_to\_BCD(time->second);

temp[1] = DEC\_to\_BCD(time->minute);

temp[2] = DEC\_to\_BCD(time->hour);

temp[3] = DEC\_to\_BCD(time->day);

temp[4] = DEC\_to\_BCD(time->date);

temp[5] = DEC\_to\_BCD(time->minute);

temp[6] = DEC\_to\_BCD(time->year);

do{

result = i2c\_ds1307\_I2CMasterSendStart(ADDRESS\_SLAVE\_DS1307, i2c\_ds1307\_I2C\_WRITE\_XFER\_MODE, 1);

}while(result != i2c\_ds1307\_I2C\_MSTR\_NO\_ERROR);

i2c\_ds1307\_I2CMasterWriteByte((uint32)0x00, 1);

for(i=0; i<=6; i++)

{

i2c\_ds1307\_I2CMasterWriteByte(temp[i], 1);

}

i2c\_ds1307\_I2CMasterSendStop(5);

}

void ds1307\_time\_init(data\_time \*time)

{

time->second = 12;

time->minute = 12;

time->hour = 12;

time->day = 1;

time->date = 29;

time->month = 8;

time->year = 22;

}

File uart\_transmitss.h

#ifndef UART\_TRANSMITSS\_H

#define UART\_TRANSMITSS\_H

#include <stdio.h>

#include "project.h"

#include "ds1307\_i2c.h"

void time\_string\_concatenation(data\_time time, char \*strings);

void test\_uart\_transmit();

#endif

File uart\_transmitss.c

#include "uart\_transmitss.h"

void time\_string\_concatenation(data\_time time, char \*strings)

{

char dayInWeek[10];

switch(time.day)

{

case Sun:

strcpy(dayInWeek, "Sun");

break;

case Mon:

strcpy(dayInWeek, "Mon");

break;

case Tue:

strcpy(dayInWeek, "Tue");

break;

case Web:

strcpy(dayInWeek, "Wed");

break;

case Thur:

strcpy(dayInWeek, "Thur");

break;

case Fri:

strcpy(dayInWeek, "Fri");

break;

case Sat:

strcpy(dayInWeek, "Sat");

break;

default:

strcpy(dayInWeek, "Error!");

}

sprintf(strings, "\n%d%c%d%c%d %s, %d%c%d%c%d",time.hour, ':', time.minute, ':', time.second, \

dayInWeek, time.date, '/', time.month, '/', 2000 + time.year );

return;

}

void test\_uart\_transmit()

{

data\_time time;

char txString[50];

time.second = 15;

time.minute = 15;

time.hour = 15;

time.day = 15;

time.date = 15;

time.month = 15;

time.year = 15;

time\_string\_concatenation(time, txString);

uart\_transmit\_UartPutString( txString );

CyDelay(1000);

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include "project.h"

#include "ds1307\_i2c.h"

#include "uart\_transmitss.h"

void system\_initialization();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

data\_time time;

char txString[50];

system\_initialization();

ds1307\_time\_init(&time);

//ds1307\_write\_data(&time);

time\_string\_concatenation(time, txString);

uart\_transmit\_UartPutString(txString);

CyDelay(1000);

for(;;)

{

Pin\_Led\_Yellow\_Write( ~Pin\_Led\_Yellow\_Read() );

ds1307\_read\_data(&time);

time\_string\_concatenation(time, txString);

uart\_transmit\_UartPutString(txString);

CyDelay(500);

}

}

void system\_initialization()

{

uart\_transmit\_Start();

i2c\_ds1307\_Start();

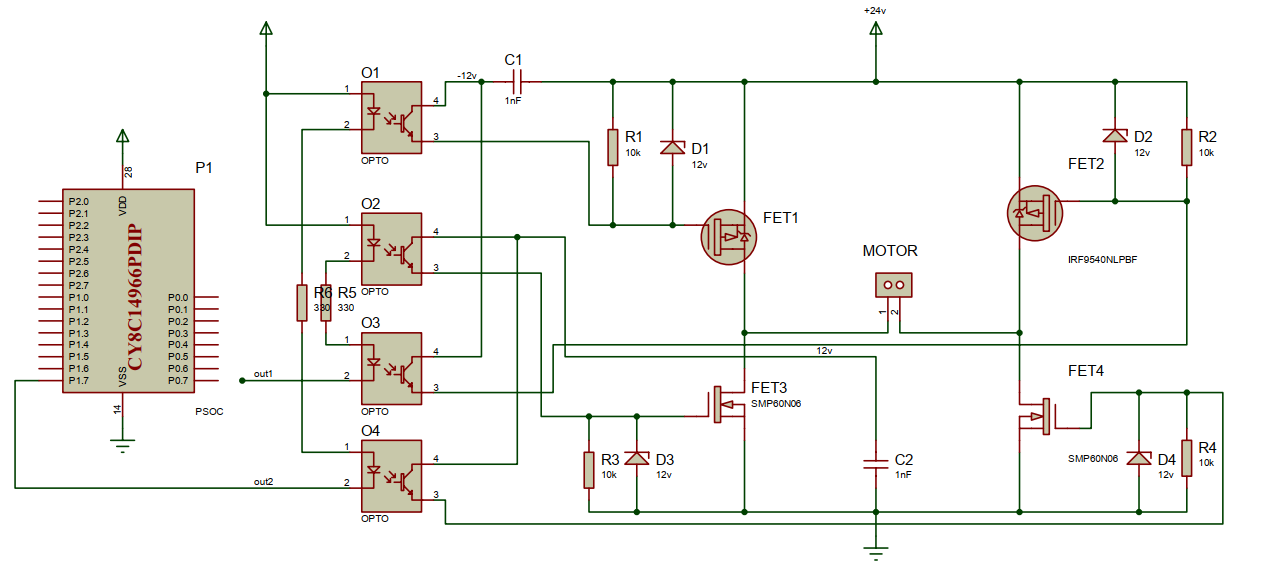
Pin\_Led\_Yellow\_Write(0);

}

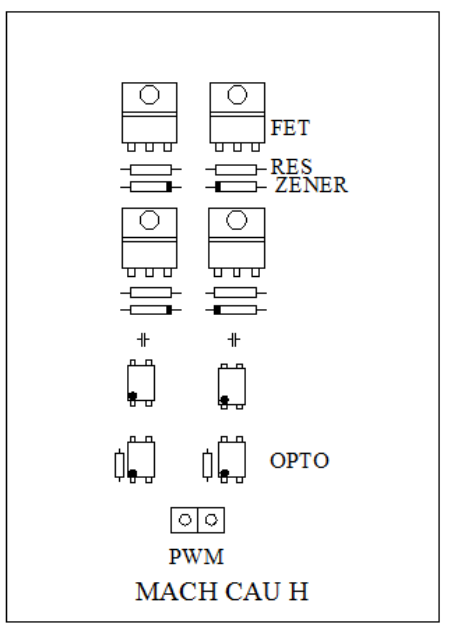
## Bài 8. Tạo xung PWM.

Tạo xung 300Hz.

1.Mạch nguyên lý.

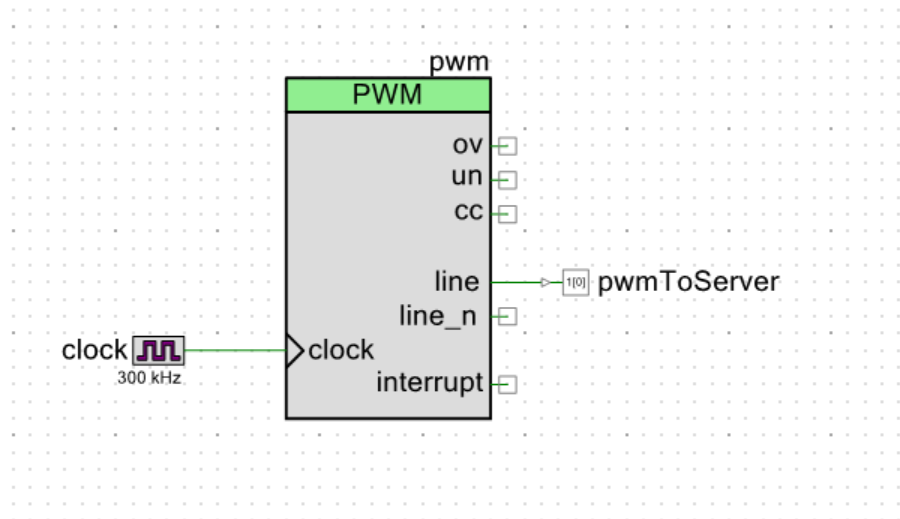


2.Mạch trên kit.

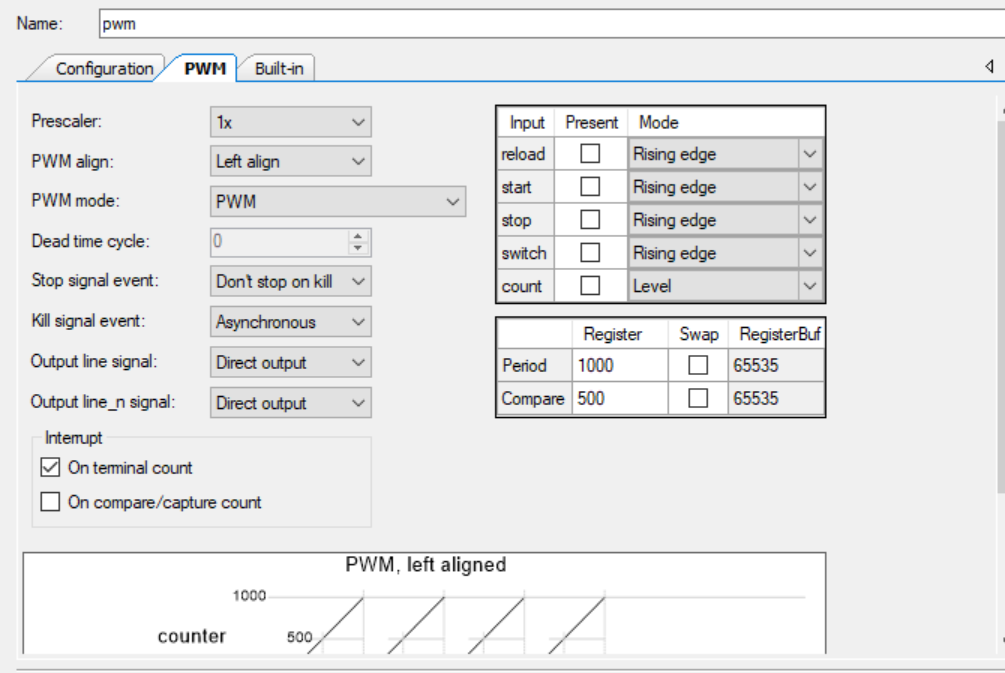


3.Kết nối cáp.





4.Viết chương trình.



File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "project.h"

#include "servo.h"

#include "ledSingle.h"

void configSystem();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

configSystem();

for(;;)

{

}

}

void configSystem()

{

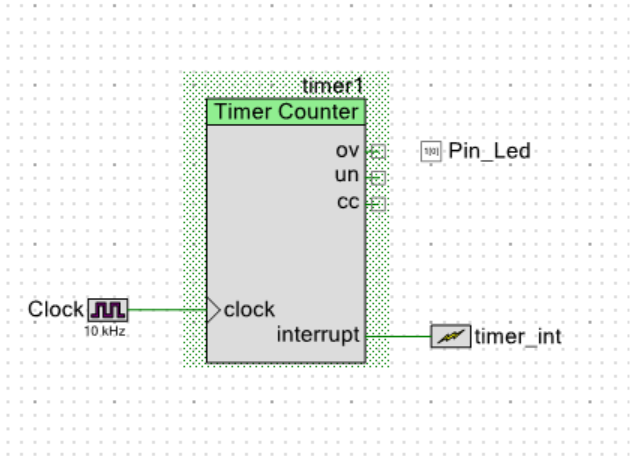
pwm\_Start();

}

## Bài 9. Thiết lập Timer và ngắt Timer.

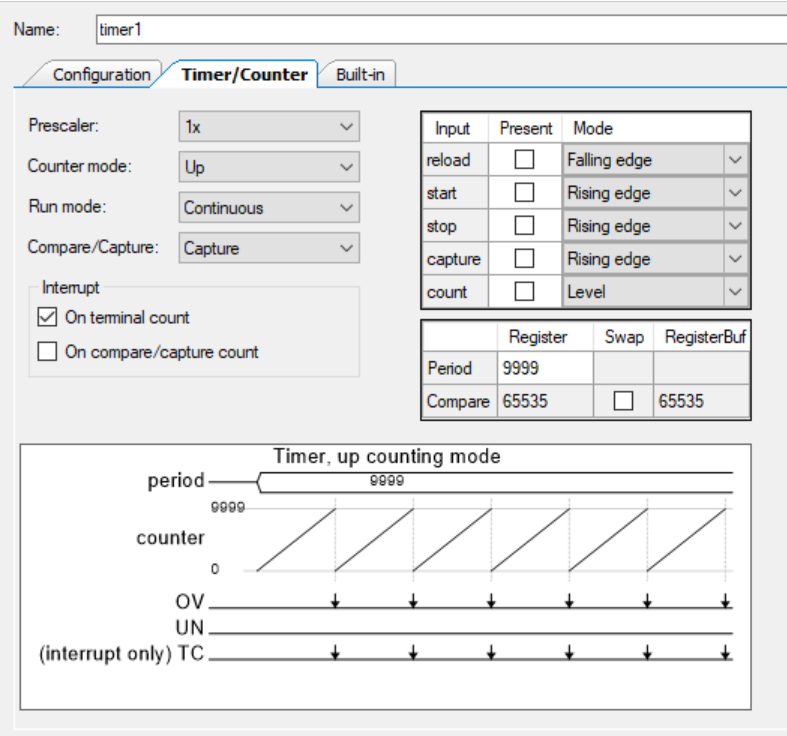
Tạo tần số ngắt 0.5 Hz, nghĩa là 1s sẽ ngắt 1 lần.

1.Kết nối cáp.





4.Viết chương trình.



File main.c

#include "project.h"

void dem\_counter();

CY\_ISR(handle\_interrupt)

{

Pin\_Led\_Write( ~ Pin\_Led\_Read());

timer1\_ClearInterrupt( timer1\_INTR\_MASK\_TC ); // clear timer

timer\_int\_ClearPending(); // clear interrupt timer

}

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

timer1\_Start();

timer\_int\_StartEx(handle\_interrupt);

for(;;)

{

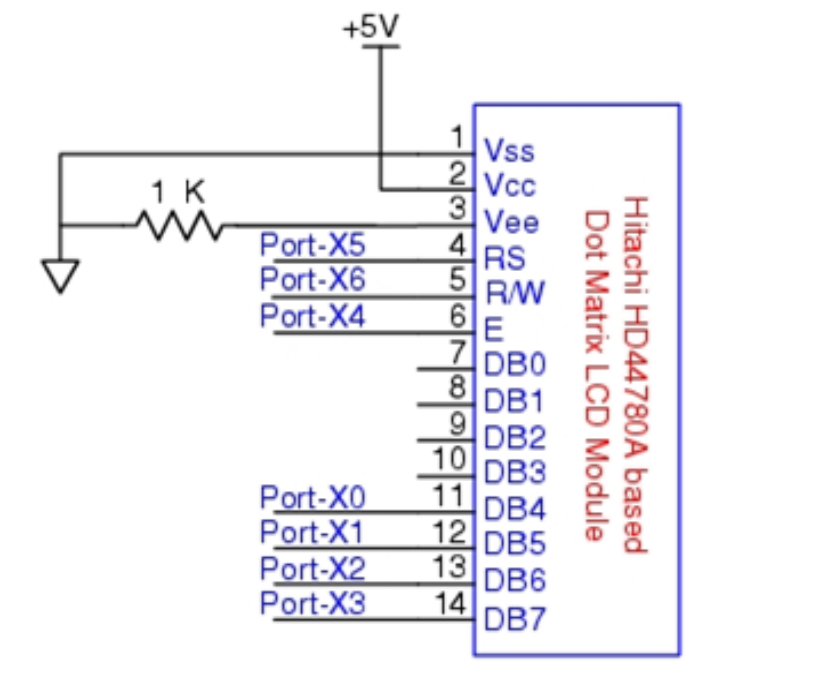
/\* Place your application code here. \*/

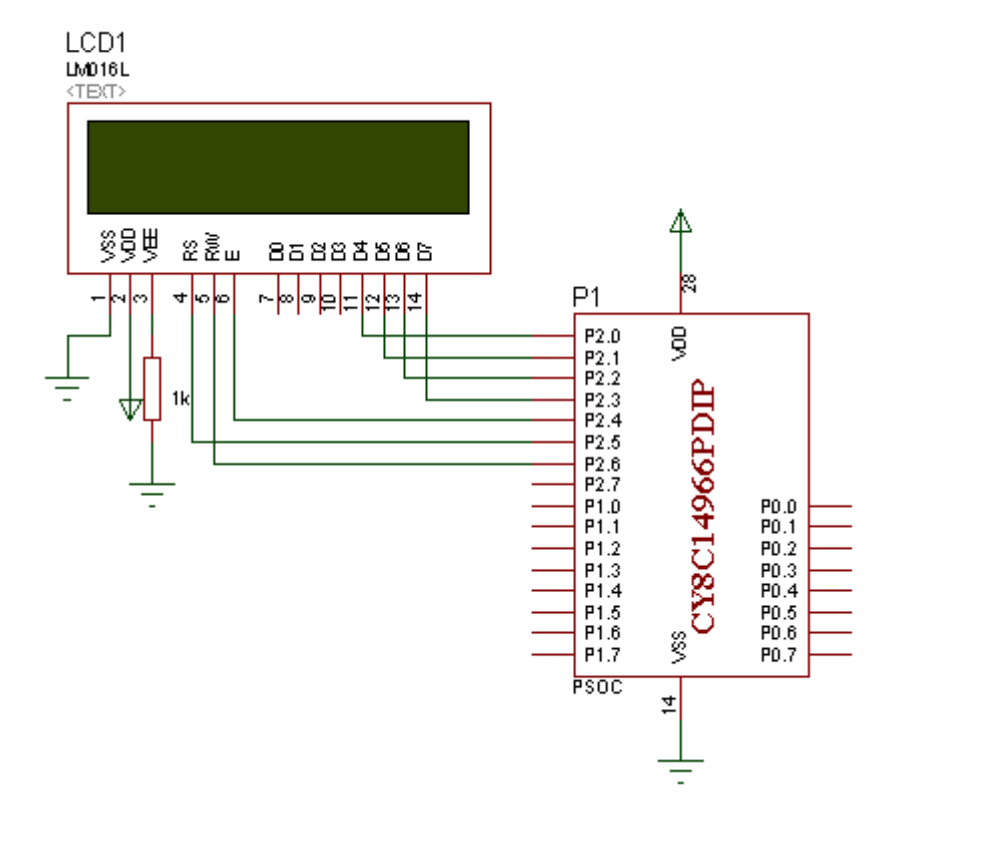
}

}

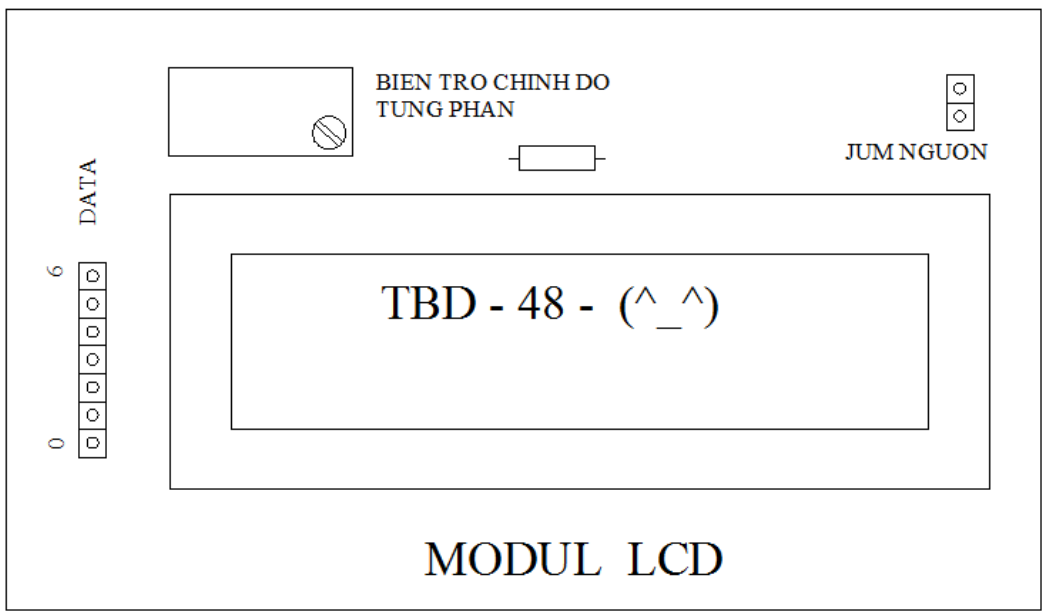
## Bài 10. Lcd.

1.Mạch nguyên lý.





2.Mạch trên kit.



3.Kết nối cáp.

Nối cáp giữa port 0 và LCD theo đúng thứ tự như số đánh trên kit.



4.Viết chương trình.

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "project.h"

void configSystem();

void display();

#endif

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

configSystem();

for(;;)

{

display();

}

}

void configSystem()

{

lcd\_Start();

lcd\_Position(0u,0u);

lcd\_PrintString("hello wourld");

lcd\_Position(1u,0u);

lcd\_PrintString("hello1 wourld");

CyDelay(1000);

lcd\_ClearDisplay();

}

void display()

{

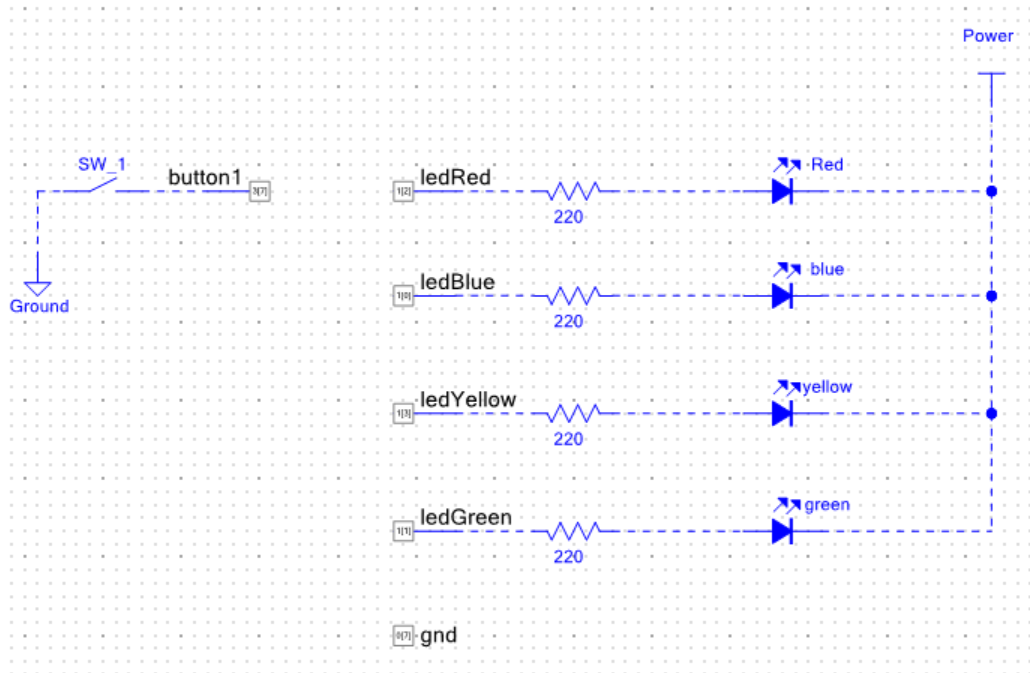
lcd\_Position(0u,0u);

lcd\_PrintString("lcd 16x2");

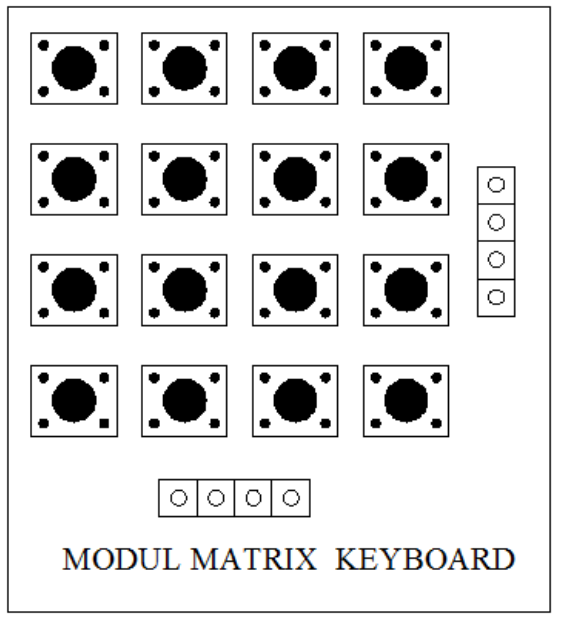
}

## Bài 11. Nút nhấn.

1.Mạch nguyên lý.



2.Mạch trên kit.



3.Kết nối cáp.



4.Viết chương trình.

File ledSingle.h

#ifndef LEDSINGLE\_H

#define LEDSINGLE\_H

#ifndef PROJECT\_H

#define PROJECT\_H

#include "project.h"

#endif

void ledTurnOff(int digit);

void ledTurnOn(int digit);

#endif

File ledSingle.c

#include "ledSingle.h"

void ledTurnOff(int digit)

{

switch(digit)

{

case 1:

ledBlue\_Write(1);

break;

case 2:

ledGreen\_Write(1);

break;

case 3:

ledRed\_Write(1);

break;

case 4:

ledYellow\_Write(1);

break;

}

}

void ledTurnOn(int digit)

{

switch(digit)

{

case 1:

ledBlue\_Write(0);

break;

case 2:

ledGreen\_Write(0);

break;

case 3:

ledRed\_Write(0);

break;

case 4:

ledYellow\_Write(0);

break;

}

}

File buttonBlack.h

#ifndef BUTTONBLACK\_H

#define BUTTONBLACK\_H

#ifndef PROJECT\_H

#define PROJECT\_H

#include "project.h"

#endif

int valuePressButton();

#endif

File buttonBlack.c

#include "buttonBlack.h"

//value return: [1,4];

int valuePressButton()

{

static int status = 0;

static int remember = 0;

if(button1\_Read() == 0)

remember = 1;

if(button1\_Read() == 1 && remember == 1)

{

remember = 0;

status++;

if( status > 4 )

status = 1;

}

return status;

}

File main.h

#ifndef MAIN\_H

#define MAIN\_H

#ifndef PROJECT\_H

#define PROJECT\_H

#include "project.h"

#endif

#include "buttonBlack.h"

#include "ledSingle.h"

void buttonHandle();

File main.c

#include "main.h"

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

gnd\_Write(0);

for(;;)

{

/\* Place your application code here. \*/

buttonHandle();

}

}

void buttonHandle()

{

ledTurnOn( valuePressButton() );

CyDelay(2000);

ledTurnOff( valuePressButton() );

CyDelay(2000);

}

# IV.Kết luận.

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

- Kit dễ sự dụng, không có sai sót trong quá trình sử dụng.

- Hệ thống bài thực hành phụ hợp với nhiều trình độ đào tạo khác nhau.

# V.Tài liệu tham khảo

Các tài liệu về hướng dẫn sử dụng, lập trình… Các project mẫu đã có do anh Khoa cung cấp.

Video hướng dẫn trên youtube:

<https://www.youtube.com/watch?v=wttaZVbmBI8&list=PLIOkqhZiy83F8KPvHejA4ujvAfwJYpAtP>