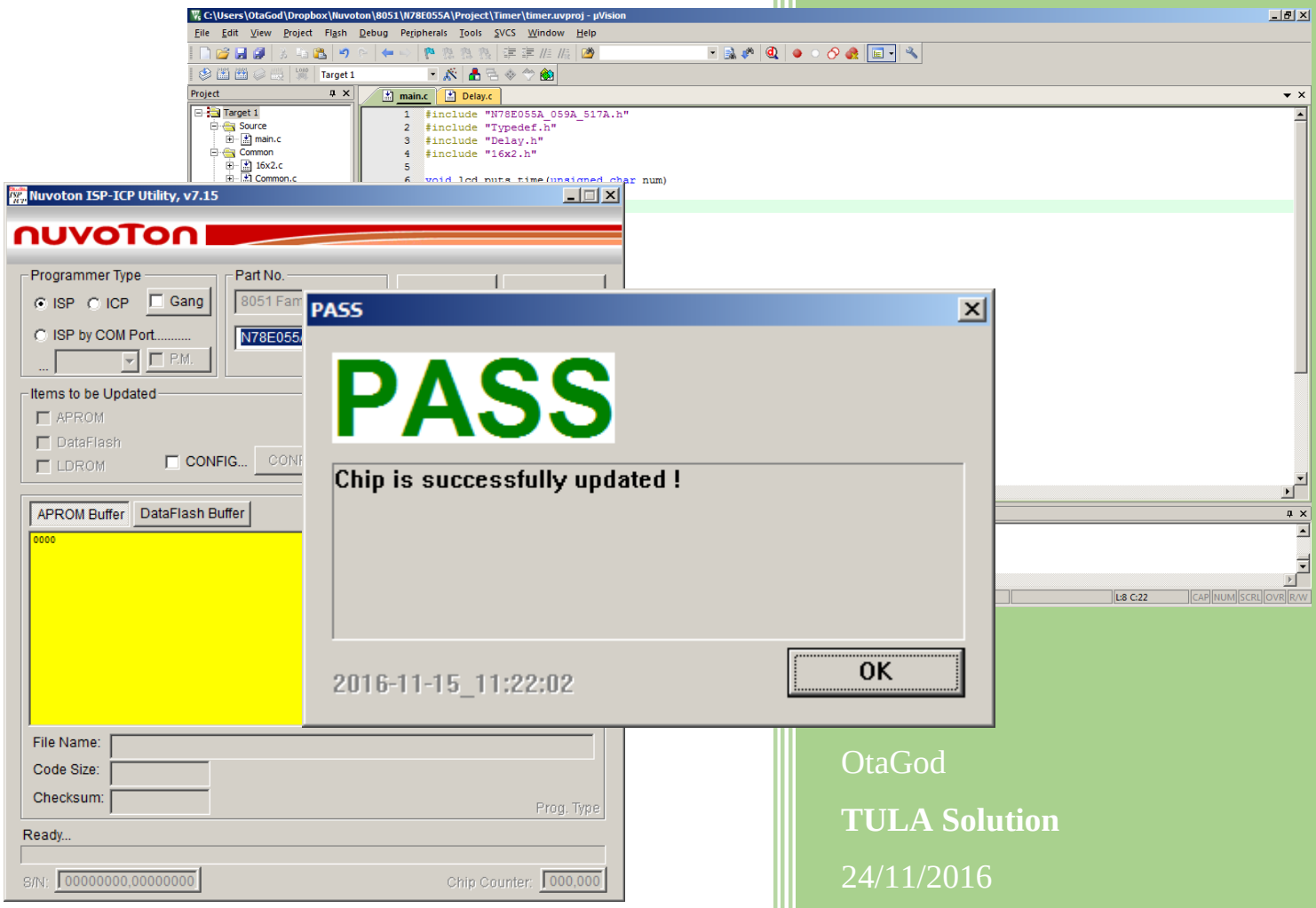
TULA Solution

The Distributor of Quality

nuvoTon

HƯỚNG DẪN PHÁT TRIỂN N78E055A TRÊN BO MẠCH AT89S52v3



OtaGod

TULA Solution

24/11/2016

Ngày: 24/11/2016

MỤC LỤC

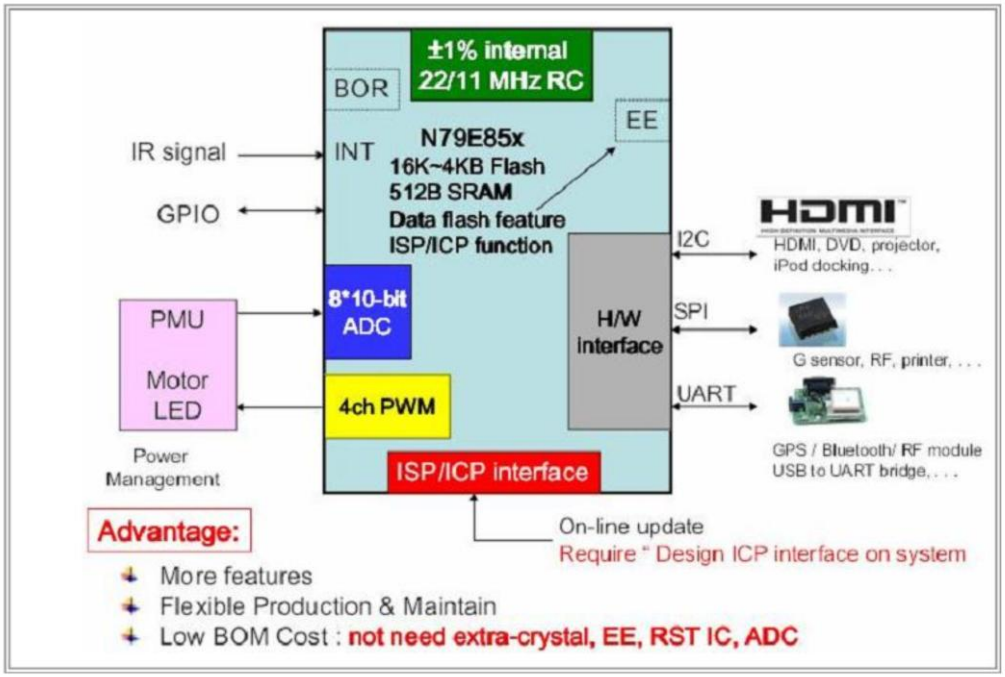
HƯỚNG DẪN PHÁT TRIỂN N78E055A TRÊN BOARD KIT AT89S52 V3

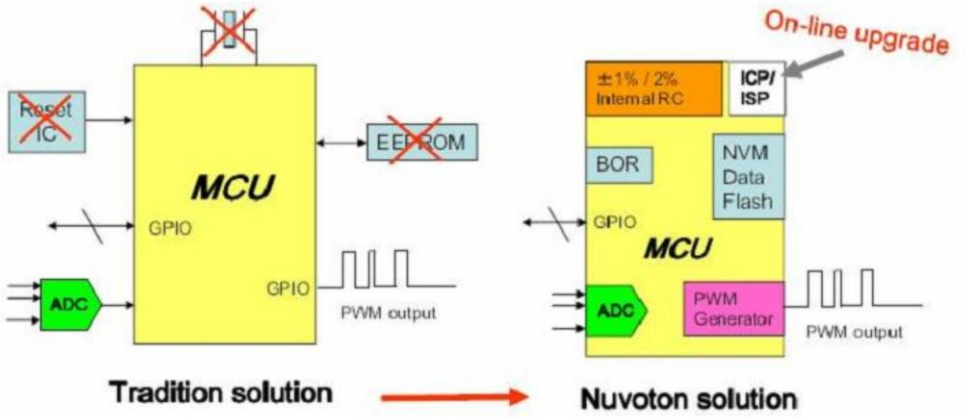
I. GIỚI THIỆU VI ĐIỀU KHIỂN N78E055A

1. Tổng Quan

N78E055A là vi điều khiển 8bit thuộc dòng 8051 của hang Nuvoton, và có thể thay thế cho các vi điều khiển dòng 8051 của hãng khác như Atmel,...

So với các vi điều khiển 8051 khác, N78E055A mang những tính năng kỹ thuật hiện đại và nổi trội như thạch anh nội tốc độ cao, tích hợp nhiều kiểu nạp, kháng nhiễu, chống ồn, giá rẻ.





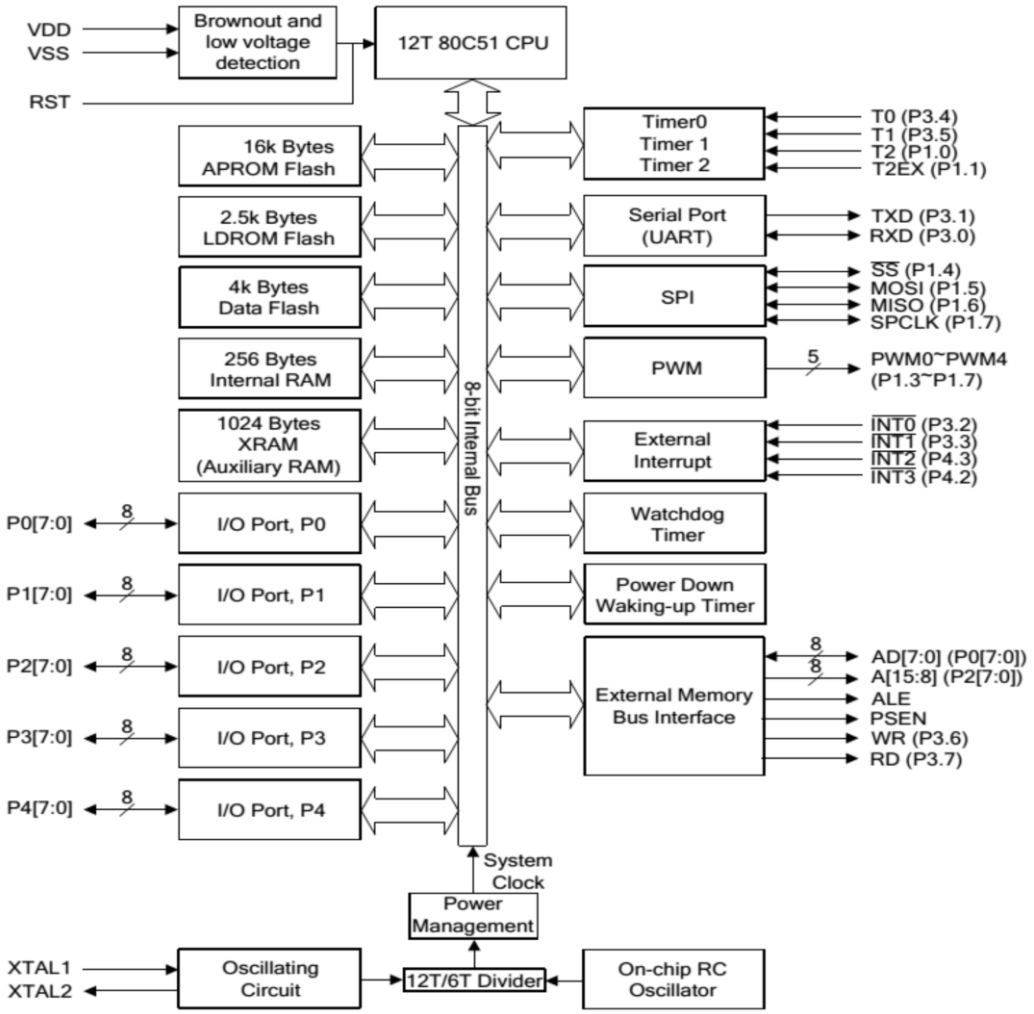
2. Tính Năng

– Dòng vi điều khiển 8bit chế tạo trên công nghệ CMOS với thiết kế chống nhiễu, tĩnh điện ESD/EFT.

- Lõi 80C51 12T/6T (chạy với 12 hoặc 6 xung dao động cho chu kỳ máy).
- Dải áp hoạt động rộng: 2.4 ~ 5.5V và tần số chạy rộng từ 4 ~ 40MHz.
- Tích hợp bộ dao động nội RC 22.1184MHz/ 11.0592MHz với sai số ±1% ở nhiệt độ 25°C và ±3% cho toàn dải nhiệt độ -40°C ~ +85°C.
- Về bộ nhớ Flash: 16KB APROM, 2.5KB ISPROM, 4KB DataFlash, với khả năng ghi xóa 10.000 lần và dữ liệu còn nguyên vẹn trên 10 năm tại nhiệt độ dưới +85°C.
- Về RAM có 255B, bổ sung thêm 1KB XRAM (dùng các lệnh MOVX để truy cập).
- Ngoài ra, còn hỗ trợ giao diện Bus bộ nhớ ngoài (Programe/Data) quản lý tới 64KB.
- Hỗ trợ nạp ISP được với dải điện áp rộng 3.0 ~ 5.0V.
- Hỗ trợ bảo mật dữ liệu.
- Ngoại vi có:
    - 3 x Timer-16bit.
    - 5 x 3 Bộ đếm / Bộ định thời 16 bit.
    - 1 x UART song công.
    - 1 x SPI.
    - 5 kênh ra PWM.
    - 11 nguồn ngắt, 4 mức ưu tiên cho mỗi nguồn.
    - 4 x Ngắt ngoài INT (dạng chân DIP40 chỉ có INT0 và INT1).
    - WDT, POR, Hỗ trợ Reset mềm, 4 mức BOD.
    - Có bộ quản lý nguồn cho chế độ Idle và Power-Down tiết kiệm điện.
- Dạng đóng gói: DIP-40, PLCC4-4, PQFP-44, LQFP-48.

Để có thể tìm hiểu rõ hơn về chức năng, cách sử dụng và thiết đặt cho con N78E055A, các bạn có thể tải datasheet tại đây.

3. Sơ Đồ Khối Chức Năng



.

II. TÀI NGUYÊN PHÁT TRIỂN

1. Phần Mềm

N78E055A của Nuvoton tương tích với tập lệnh 8051 chuẩn (MCS-51) cho nên mọi môi trường phát triển, trình biên dịch, phần mềm mô phỏng dùng được cho chip MCU 8051 chuẩn thì đều dùng được cho chip dòng này. Ví dụ các phần mềm Keil C (uVision), IAR, Hi-Tech, SPKT-C (SPKT-8051),… Trong tài liệu này sử dụng phần mềm Keil C.

Với thiết kế mạch (PCB) thì hoàn toàn tương tự như với dòng 8051 của các hãng khác. Các phần mềm thiết kế PCB thông dụng: Altium Designer, Orcad, Eagle, Proteus,…

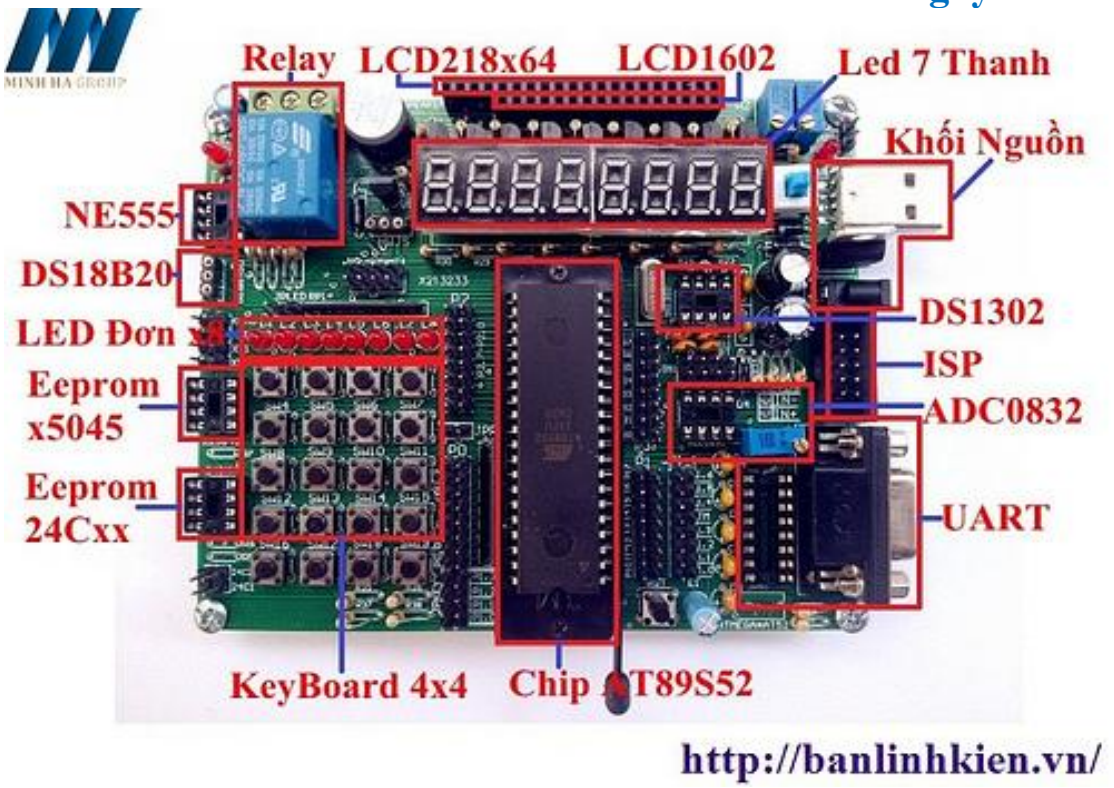
2. Phần Cứng

Công cụ nạp (mạch nạp): N78E055A được hỗ trợ giải thuật nạp bởi hầu hết các máy nạp rom đa năng của các hãng sản xuất bộ nạp ROM nổi tiếng trên thế giới như Xeltek, Elnec, Hilosystems, Leap Electronics…

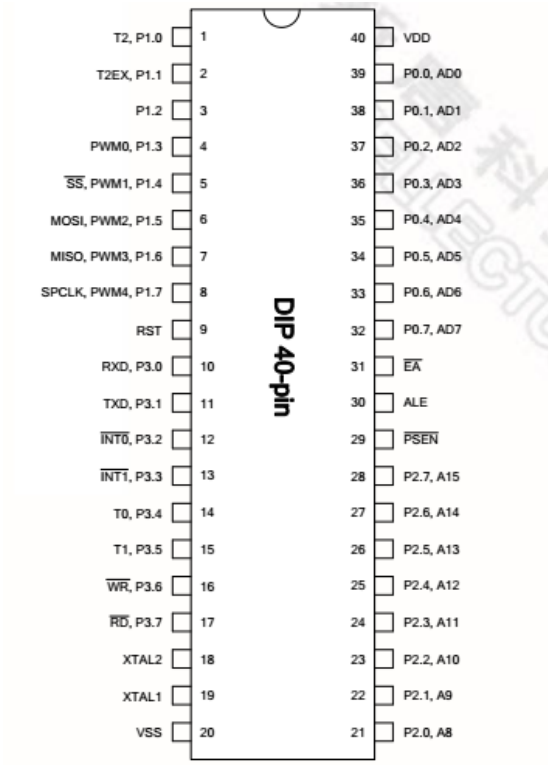
Ngoài ra hãng Nuvoton cũng chế tạo riêng các mạch nạp có kích thước nhỏ gọn, rẻ tiền mà hỗ trợ khả năng nạp được toàn bộ các chip MCU lõi 8051 của hãng, giúp tiện dùng cả trong quá trình phát triển lẫn sản xuất. Mạch nạp Gang của Nuvoton cho sản xuất hàng loạt. Mạch nạp Stand-alone kiểu ISP/ICP:

Tài liệu tập trung vào việc phát triển vi điều khiển N78E055A của Nuvoton trên board mạch thông dụng kit AT89S52 v3 của Minh Hà Group. Bộ kit tích hợp nhiều phần cứng cơ bản, giúp người mới dễ dàng trong học tập và ứng dụng sau này như: Led 7 đoạn, lcd 16x02, ma trận phím 4x4, kết nối UART,...

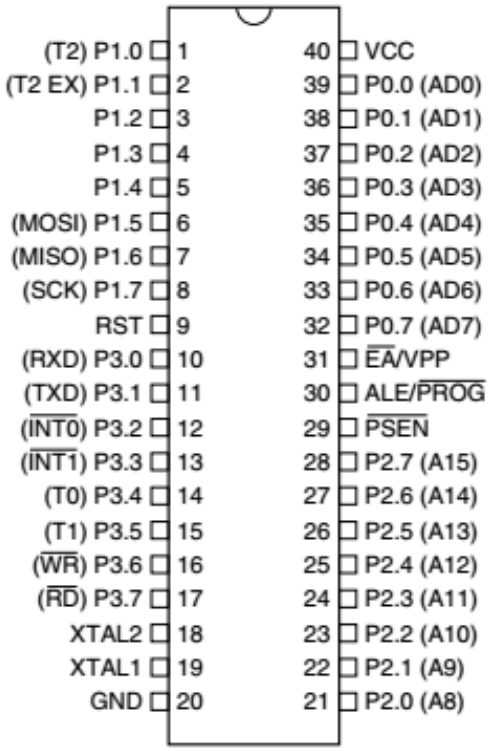
Có thể tải mạch nguyên lý của board kit tại đây.



So sánh chân của vi điều khiển N78E055A (bên trái) và AT89S52 (bên phải). Số lượng, vị trí, chức năng của chân 2 vi điều khiển hầu như là giống nhau, chính vì thế, ta có thể dễ dàng phát triển N78E055A trên board kit của AT89S52.



N78E055A pin configuration.

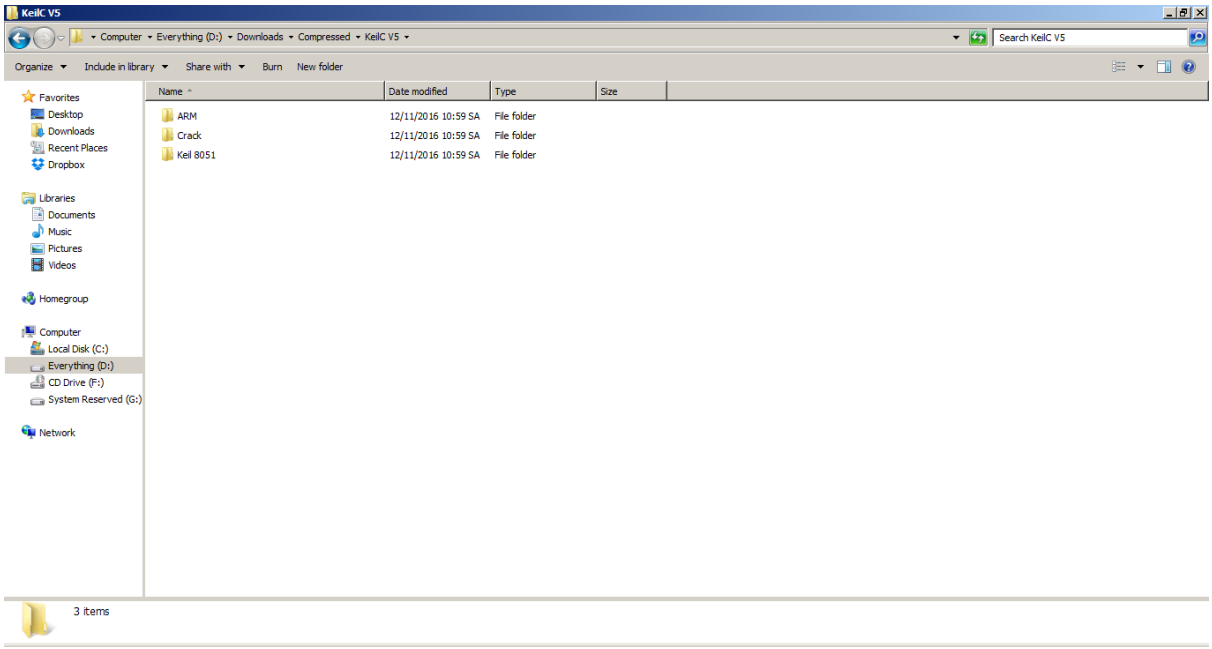


AT89S52 pin configuration.

III. HƯỚNG DẪN CÀI ĐẶT PHẦN MỀM KEIL C

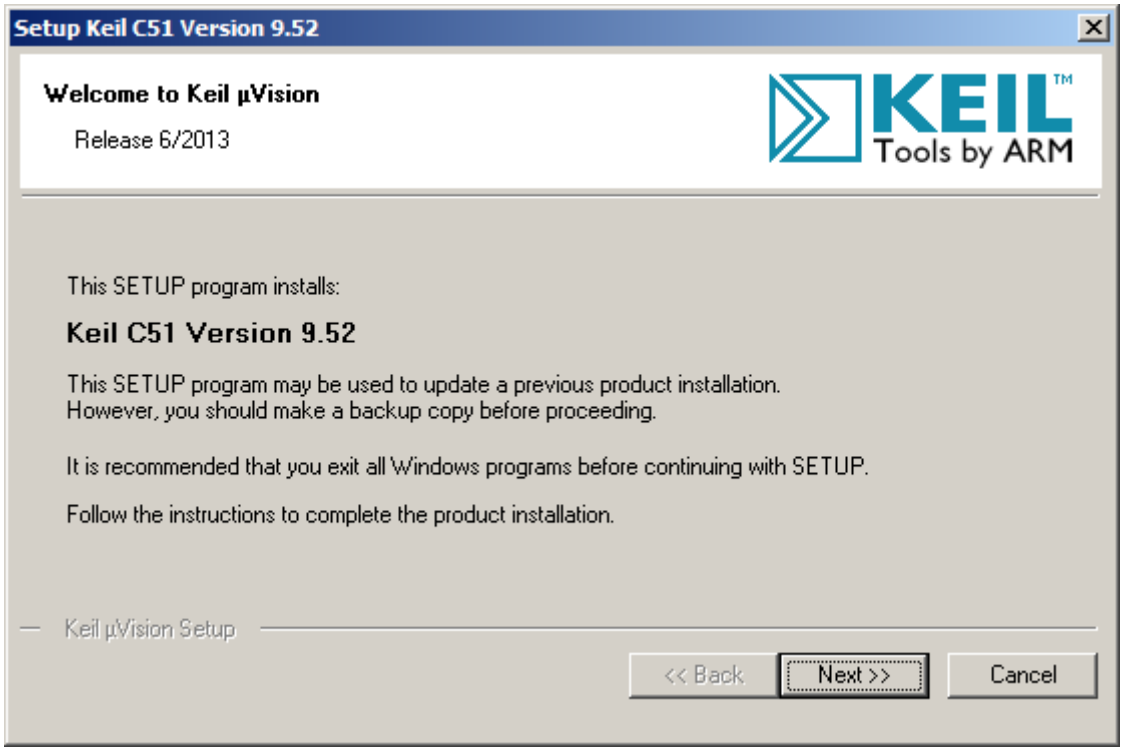
B1: Tải phần mềm Keil C tại *https://www.fshare.vn/file/QQGEGWK2D6/*

B2: Giải nén tập tin vừa tải về:

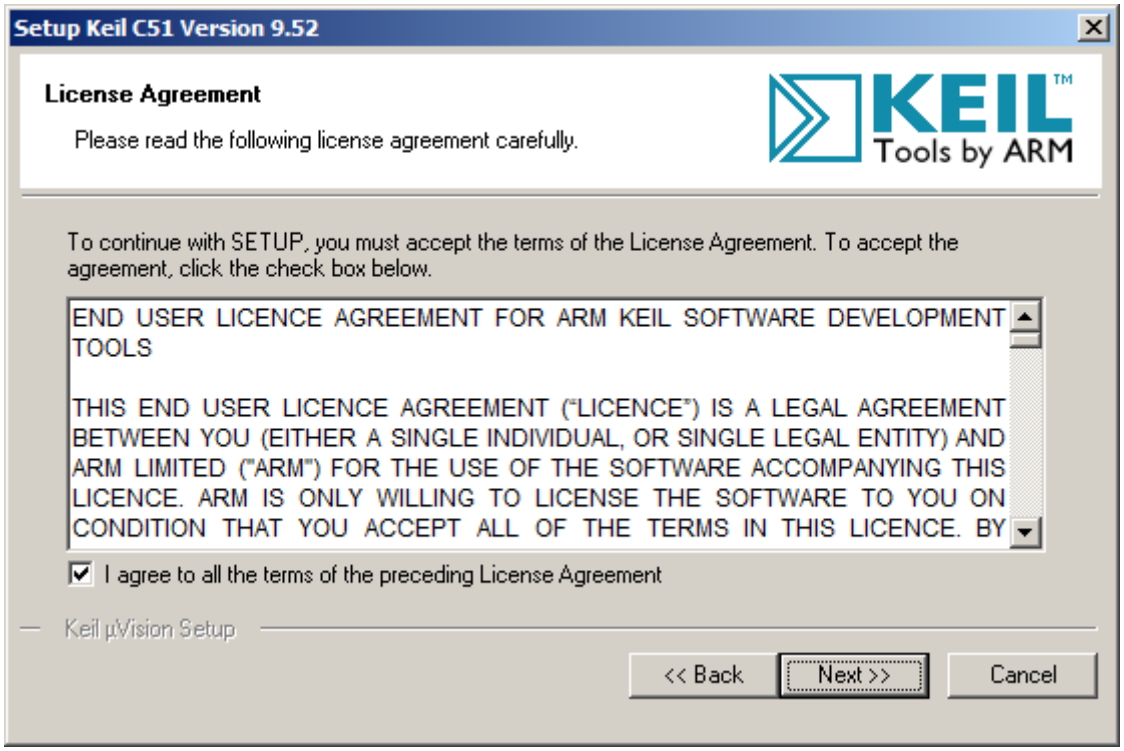


B3: Chạy file *c51v952.exe* trong thư mục …/Keil 8051.

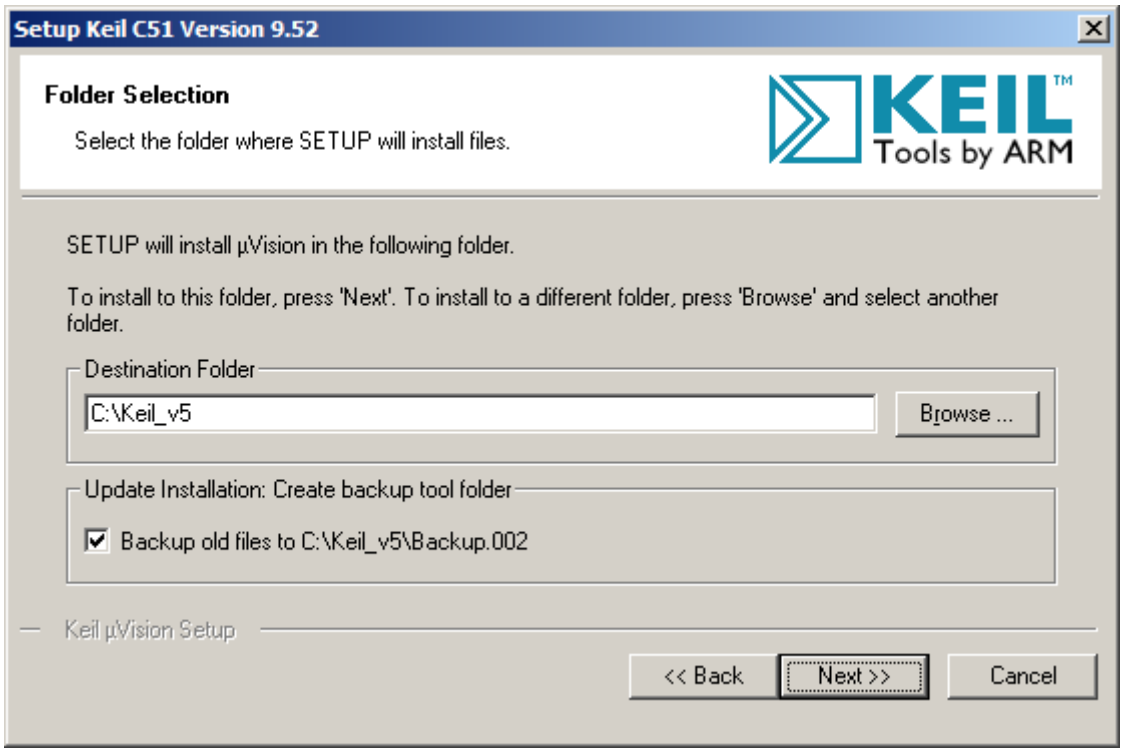
- Chọn Next:



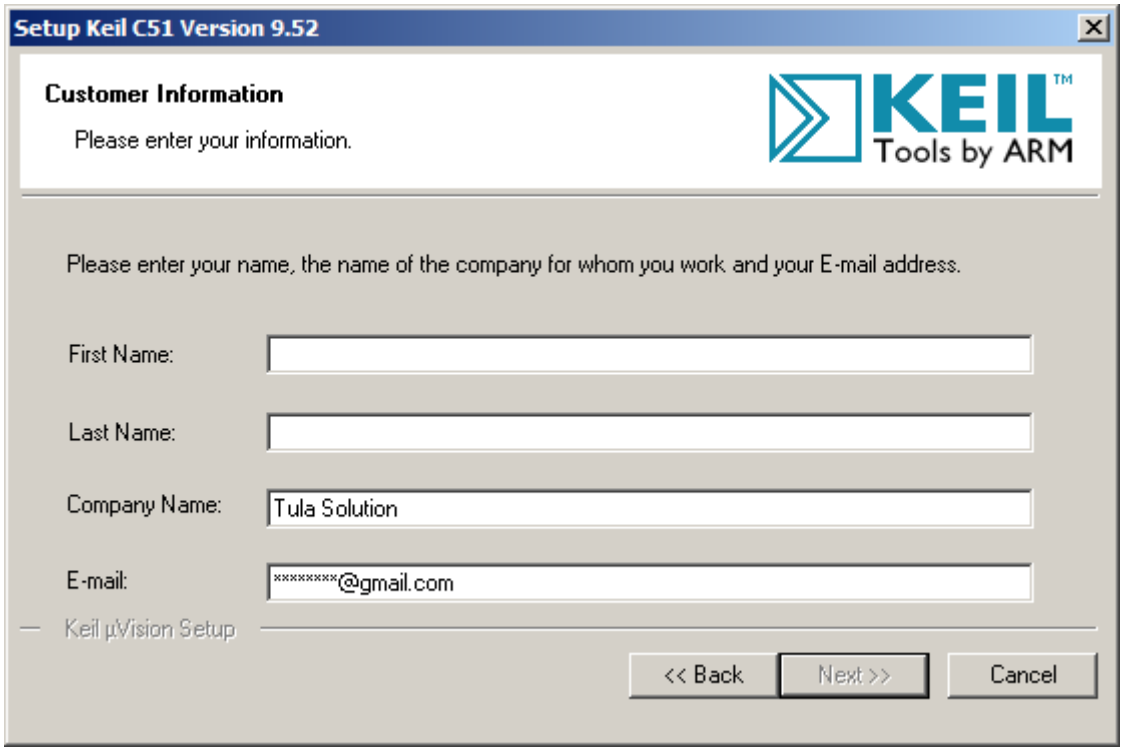
- Tích vào "*I agree to…*" và chọn Next:



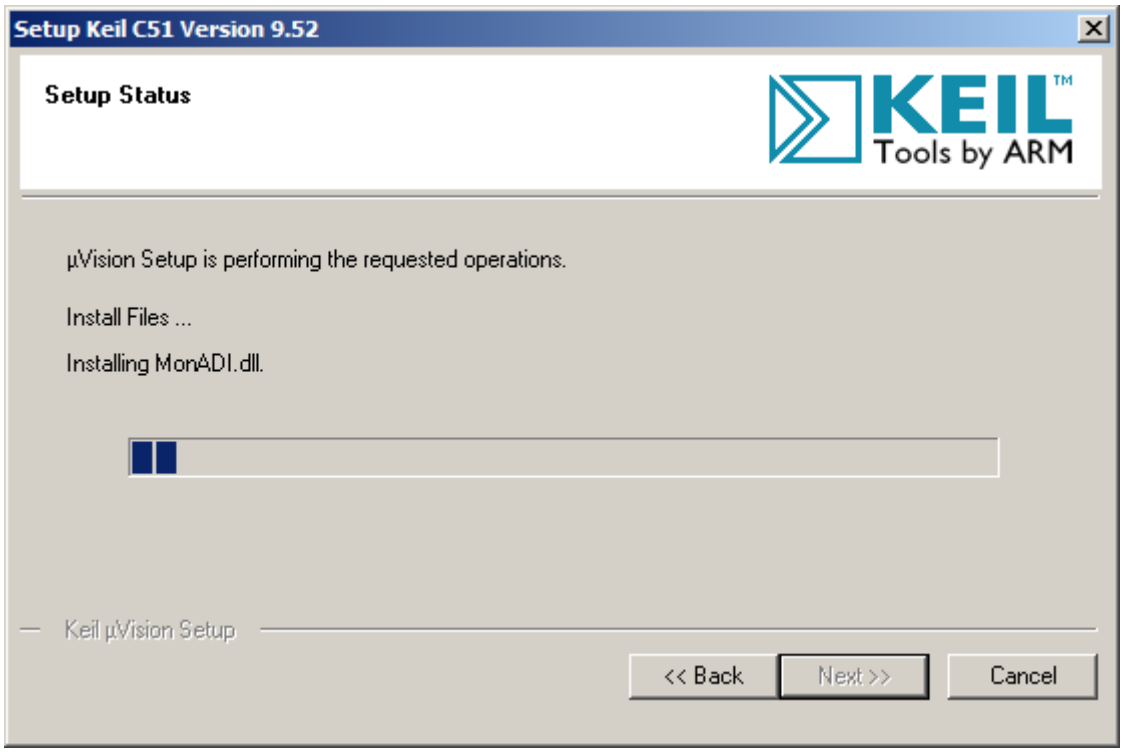
- Chọn thư mục chứa chương trình Keil C:



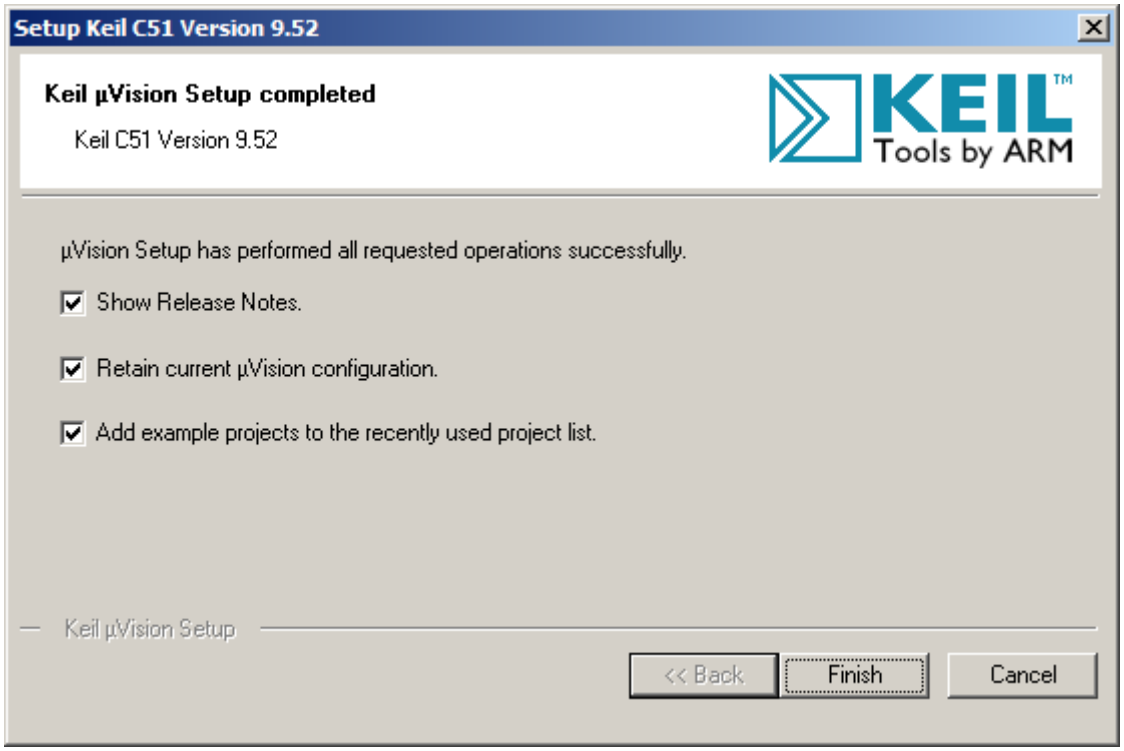
- Điền thông tin cần thiết, sau đó chọn Next:



- Đợi chương trình cài đặt hoàn thành.

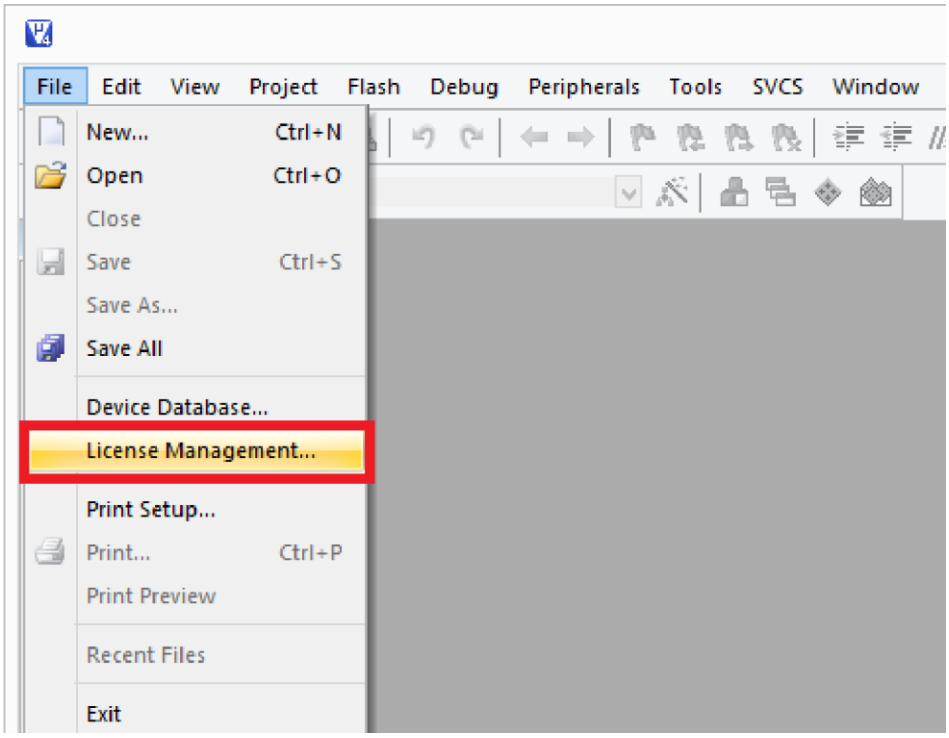


- Cài đặt xong, ta chọn Finish:

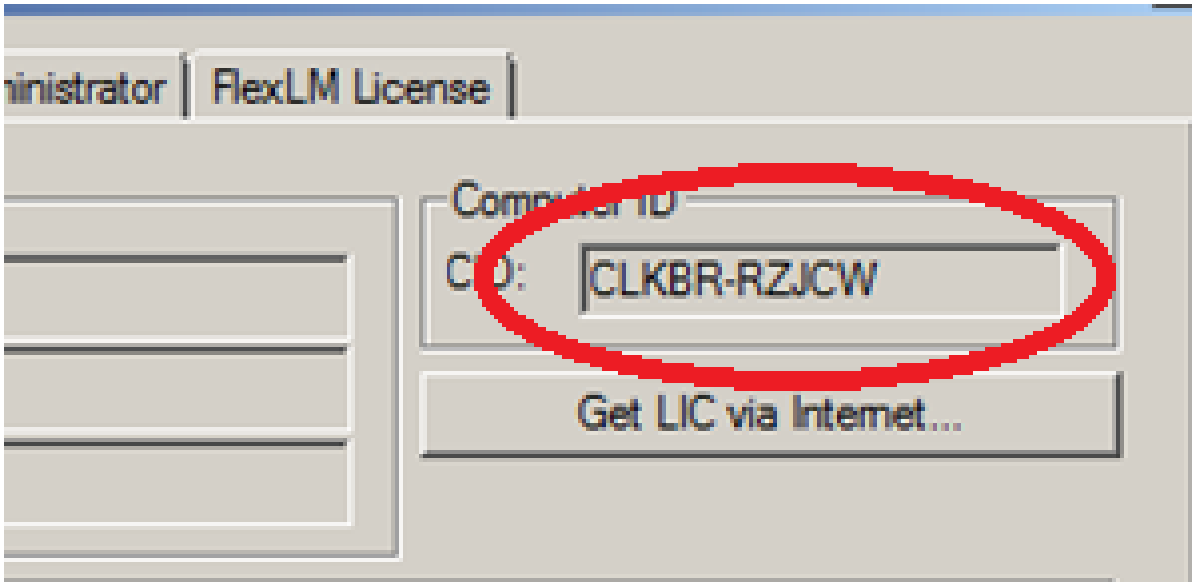


B4: Crack phần mềm Keil C.

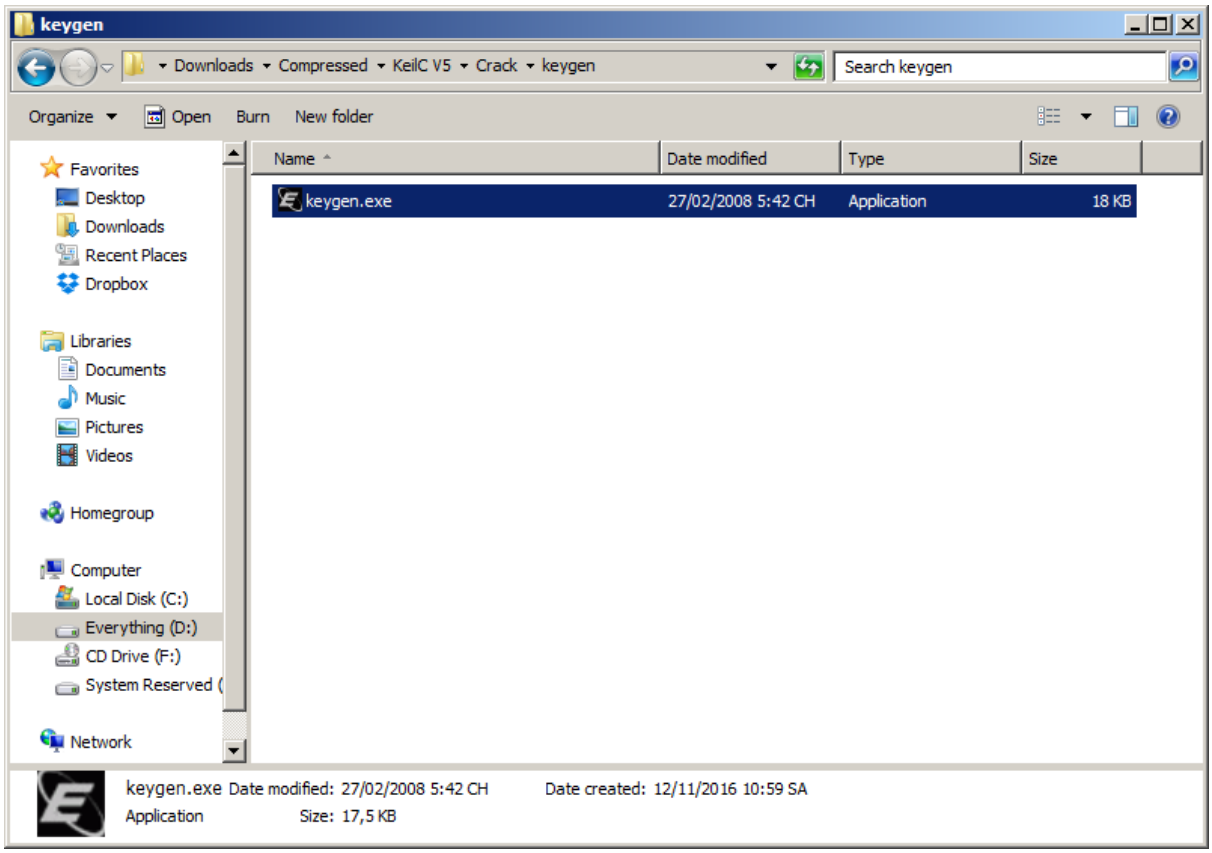
- Sau khi cài đặt xong phần mềm, ta khởi động chương trình bằng shortcut ngoài màn hình.
- Vào thanh công cụ File và chọn License Management…



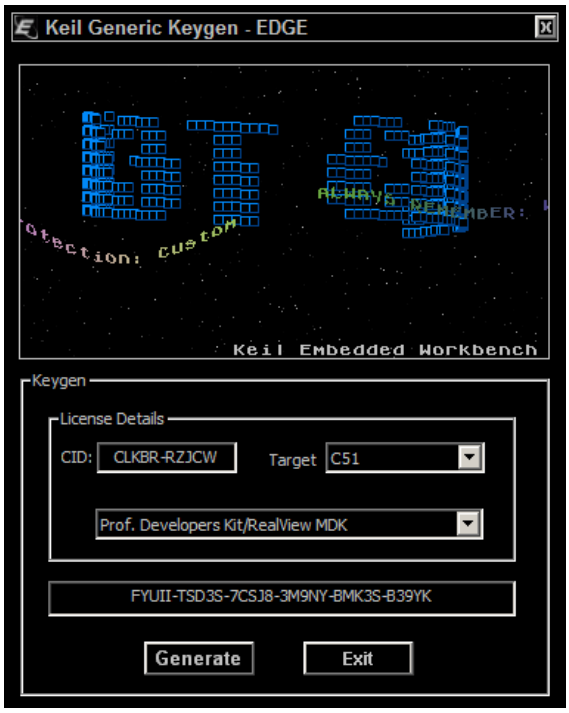
- Coppy đoạn mã code CID:



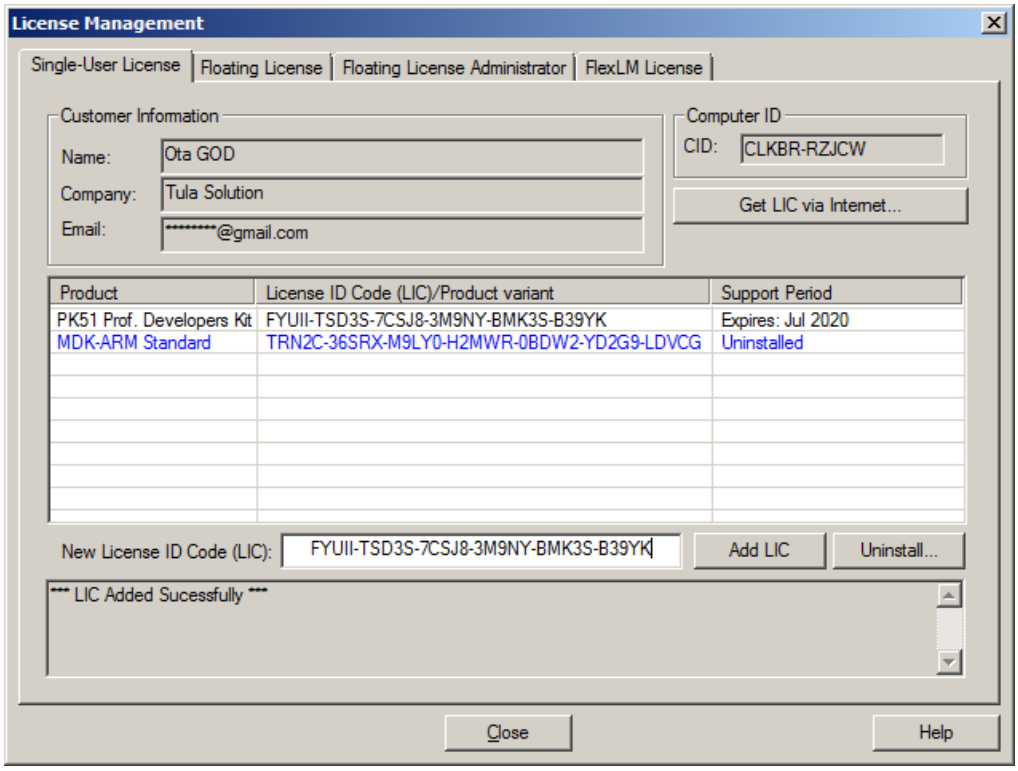
- Chạy file keygen.exe trong thư mục "…/crack/keygen" mà ta giải nén ở B2:



- Paste đoạn CID code và chọn như hình, sau đó chọn Generate:



- Coppy đoạn code ở dòng thứ 3 và vào lại License Management của phần mềm Keil C.
- Paste mã code vừa coppy vào dòng New license ID code và chọn Add LIC.



Như thế ta đã crack xong phần mềm Keil C, thế nhưng, để có thể sử dụng phần mềm này ta cần có thêm 8051 driver để cài thêm thư viện cho chip của nuvoton và sample code để thuận tiện cho công việc lập trình.

Có thể tải Nuvoton 8051 driver dành cho Keil phiên bản mới nhất tại trang chủ của Nuvoton hoặc phiên bản v1.08.

- *http://www.nuvoton.com/*
- *https://www.fshare.vn/file/83HUK17NEOZP*

Giải nén và chạy file:

Nuvoton_8051_Keil_uVision_Driver_v1.08.6426.exe

Các bước cài đặt giống như cài đặt phần mềm Keil C, chú ý thư mục cài đặt phải là thư mục cài đặt của phần mềm Keil C.

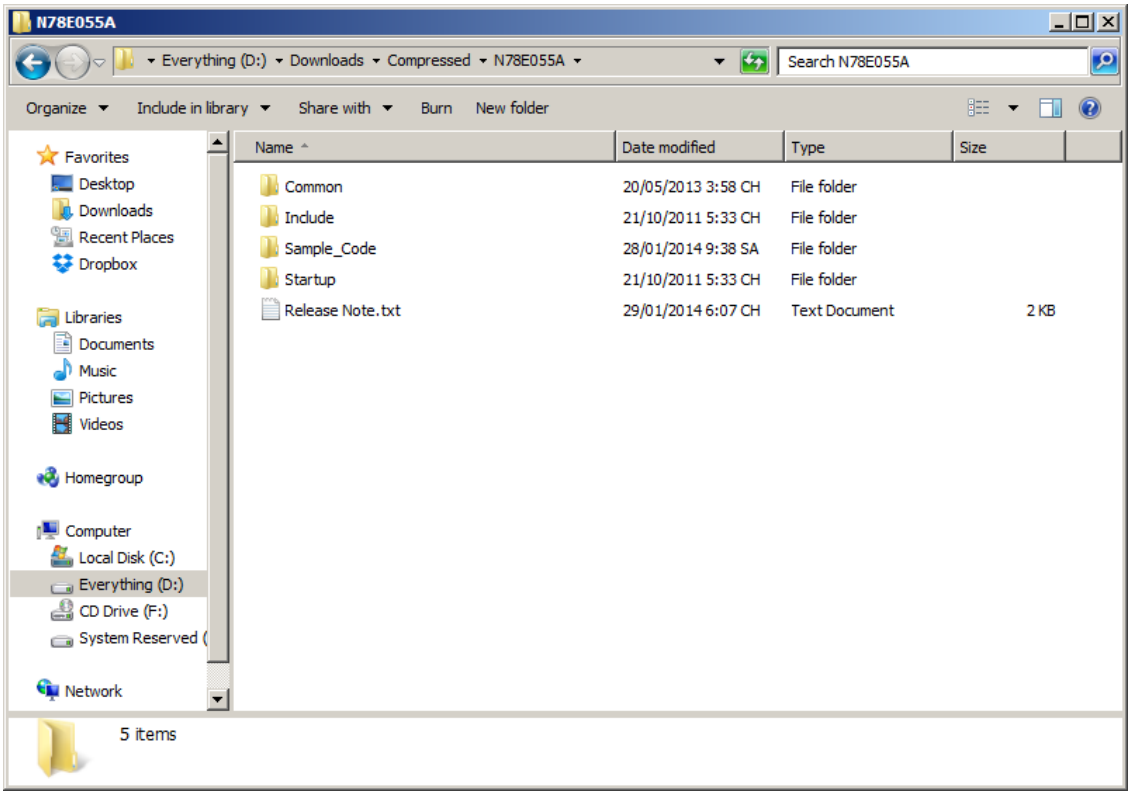
Để thuận tiện cho việc lập trình N78E055A ta cần sample code do hãng Nuvoton cung cấp, có thể tải trên trang chủ Nuvoton hoặc phiên bản v1.02.

- *https://www.fshare.vn/file/L5ORIZ6MDO32*

Sau khi tải xong sample code, ta giải nén ra thư mục dùng để lập trình.

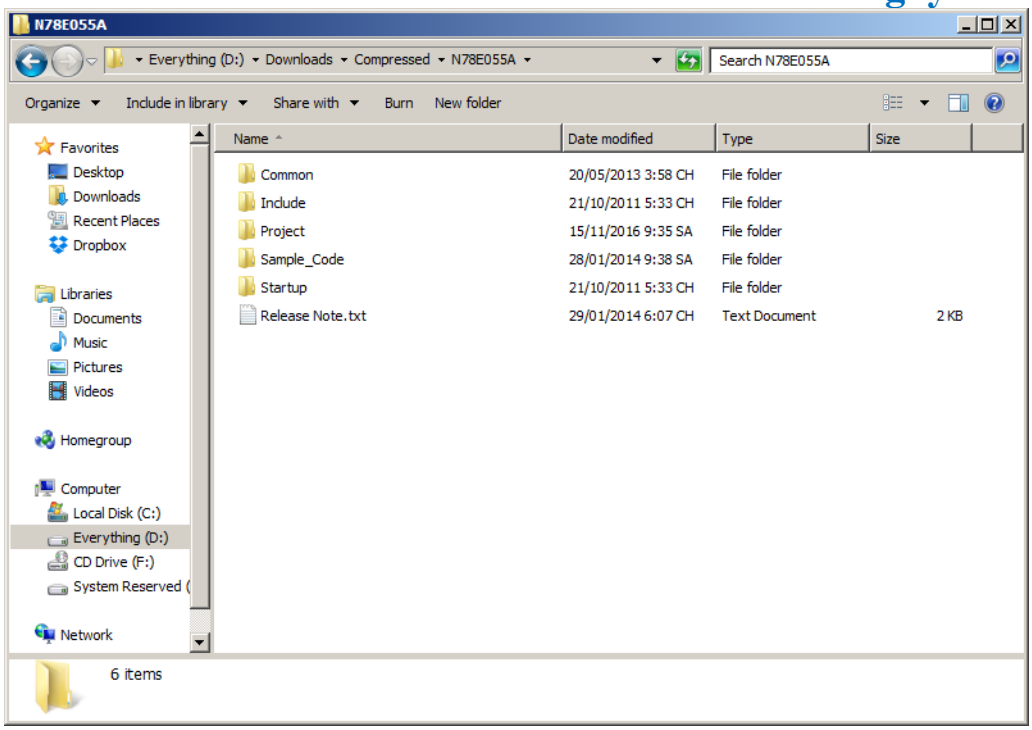
- Ví dụ: *.../N78E055A/*

Trong đó chứa các thư mục con: Common, Startup, Include, Sample code

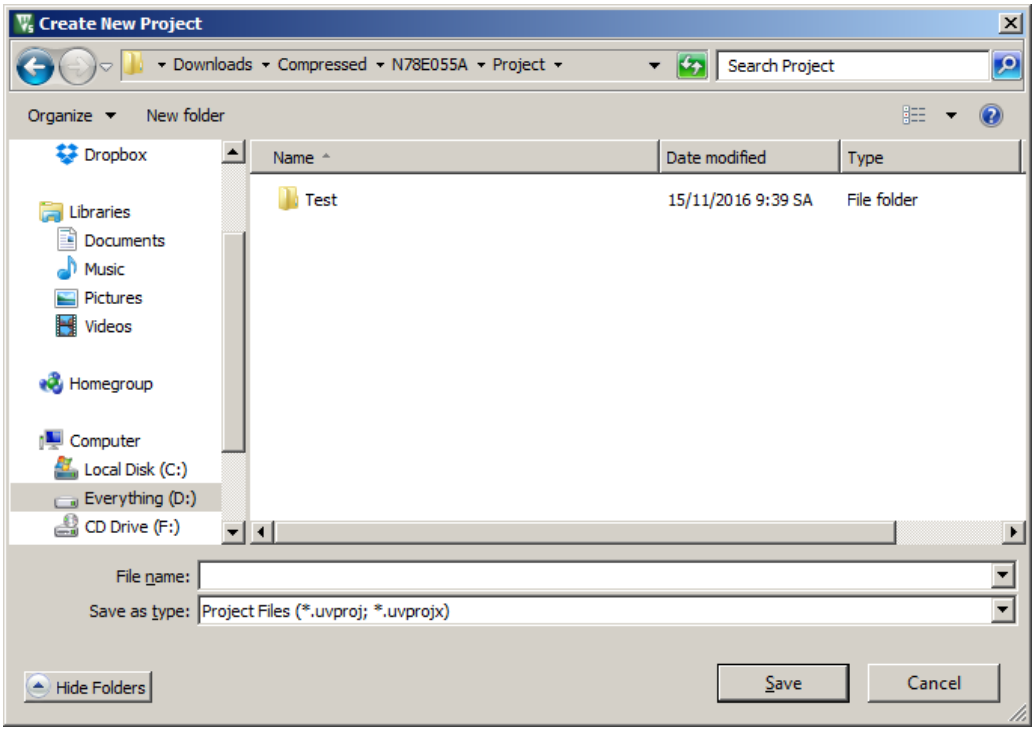


IV. HƯỚNG DẪN BUILD MỘT PROJECT N78E055A

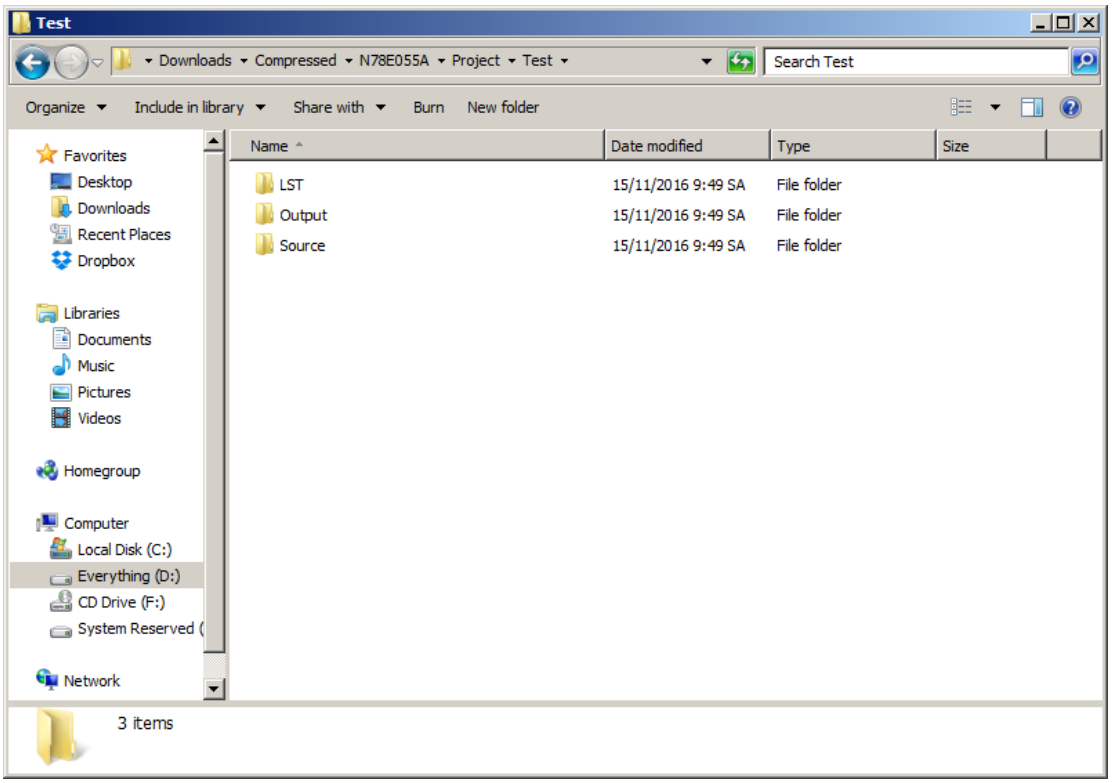
Ta sẽ sử dụng luôn thư mục …/N78E055A chứa các thư mục con khi giải nén sample code. Để dễ dàng cho việc quản lý, có thể tạo thêm một thư mục project để chứa các code.



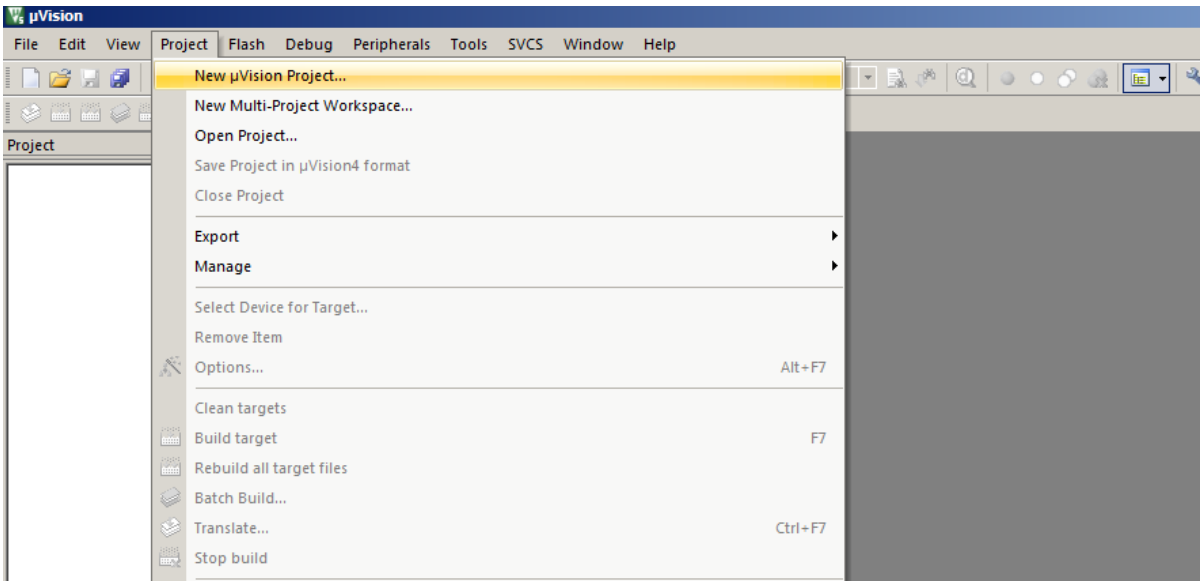
- Tạo thư mục con trong thư mục Project …/Project/Test.



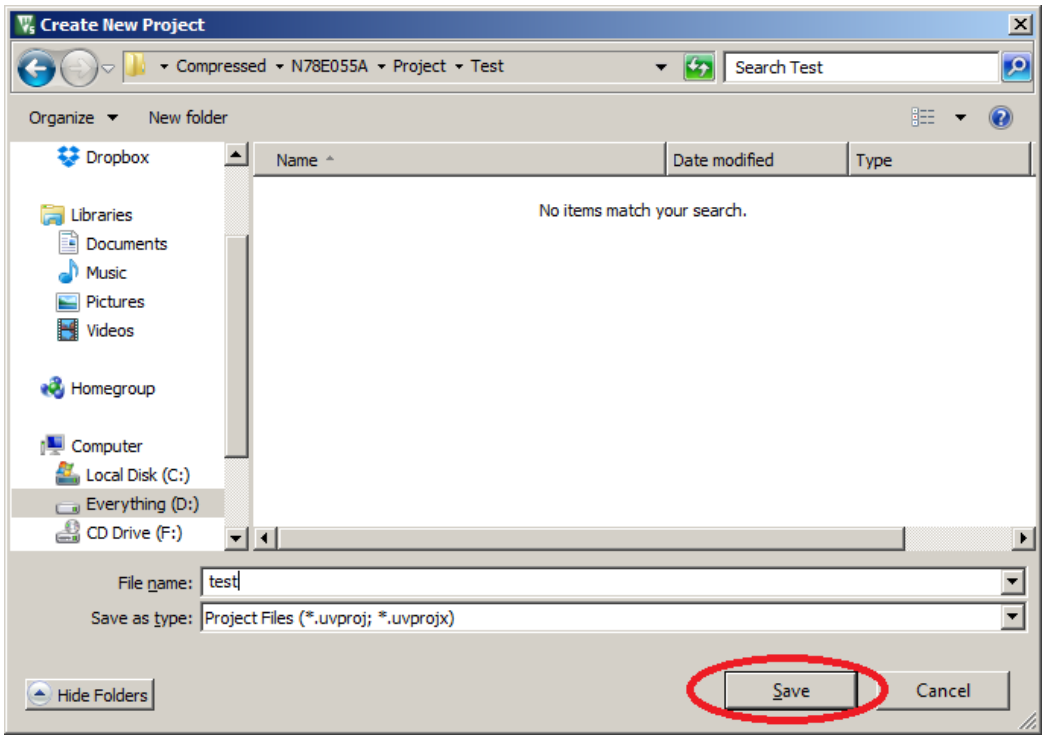
- Tạo các thư mục Output, Source, LST trong thư mục Test



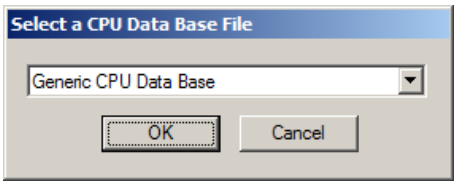
- Ta chạy phần mềm Keil C ngoài màn hình Desktop, vào thanh công cụ Project và chọn New uvision project



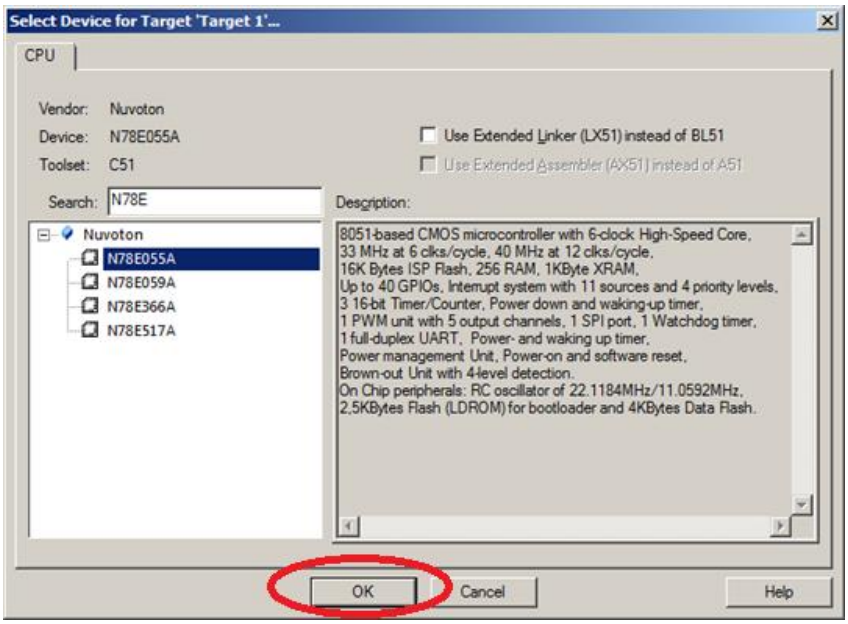
- Đặt tên project là test và save trong thư mục Test



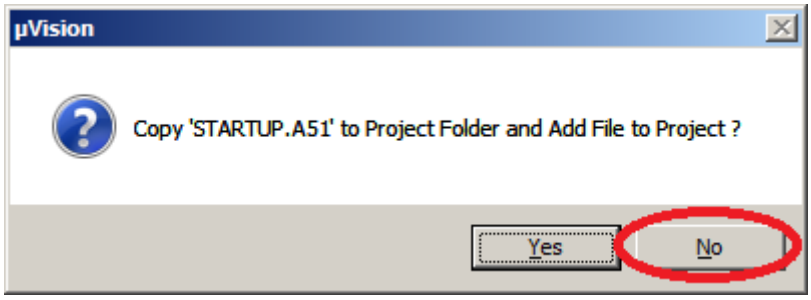
- Chọn OK



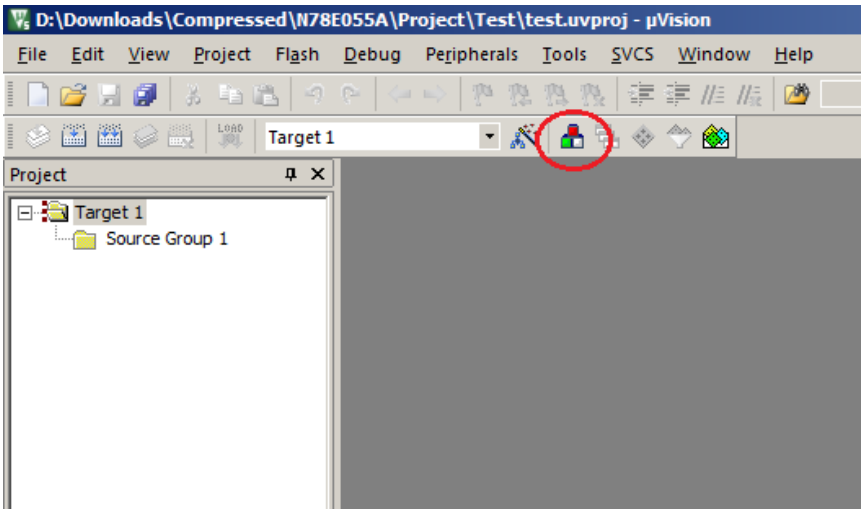
- Tìm và chọn chip N78E055A



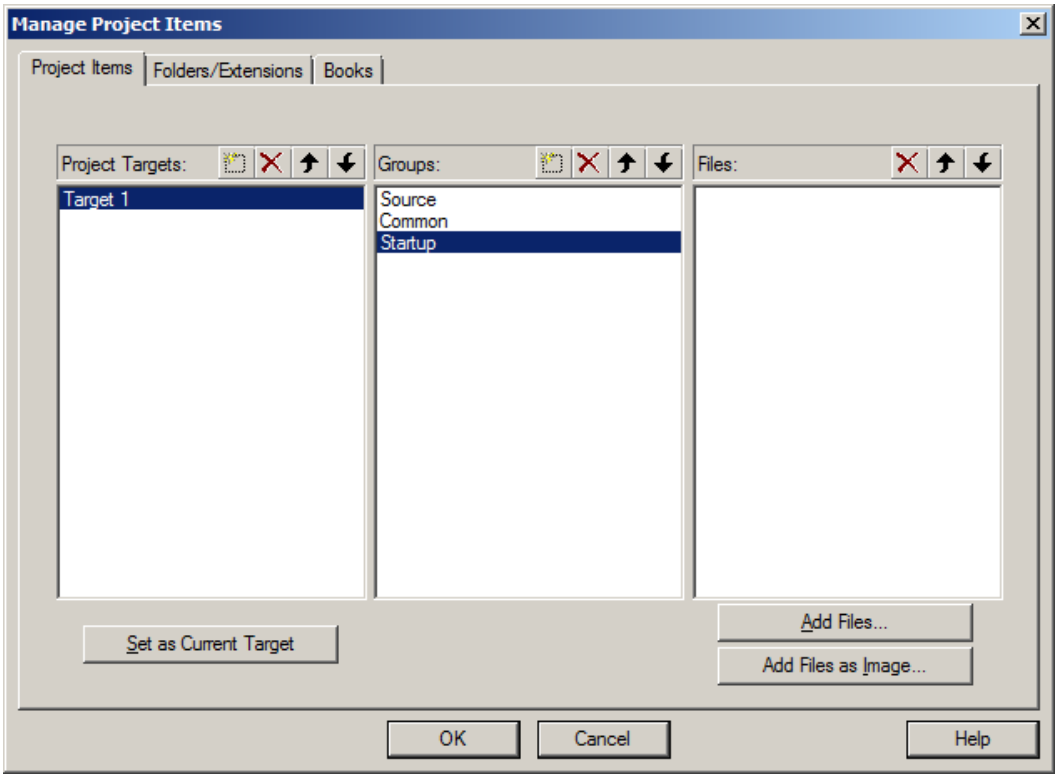
- Chọn No



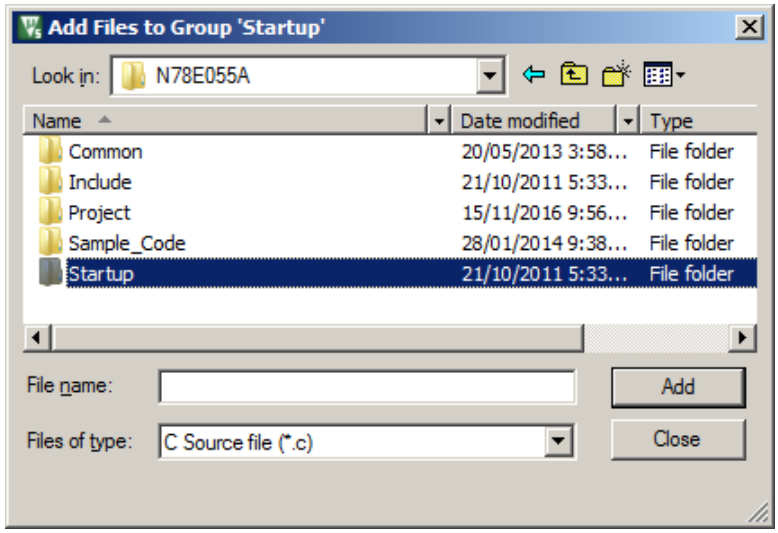
- Chọn Manage Project item trên thanh công cụ



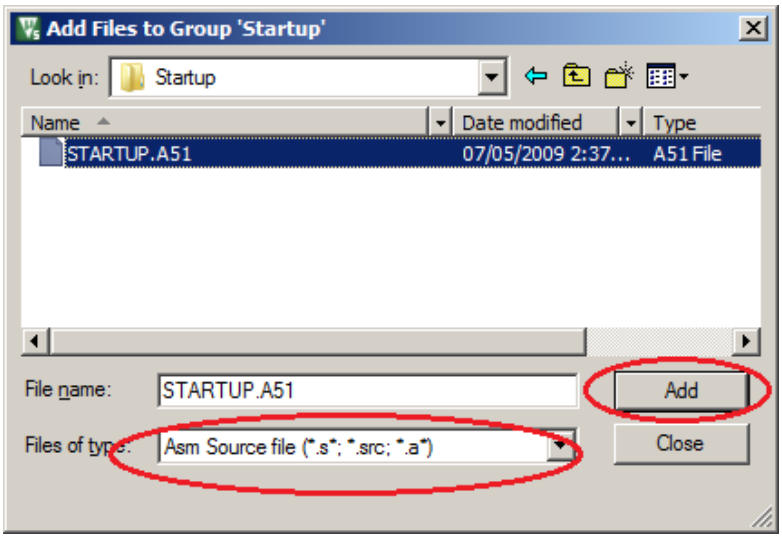
- Tạo thêm group Source, Common, Startup



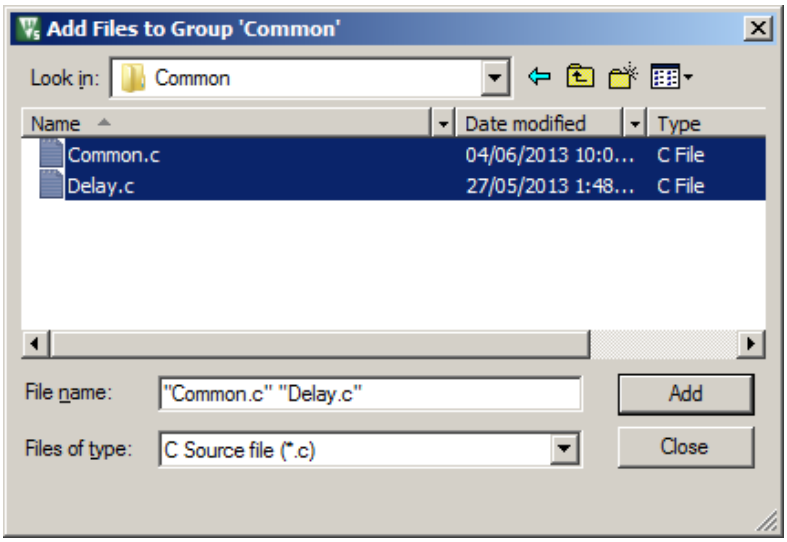
- Chọn Startup và chọn Addfile, chuyển đến địa chỉ …/N78E055A/Startup:



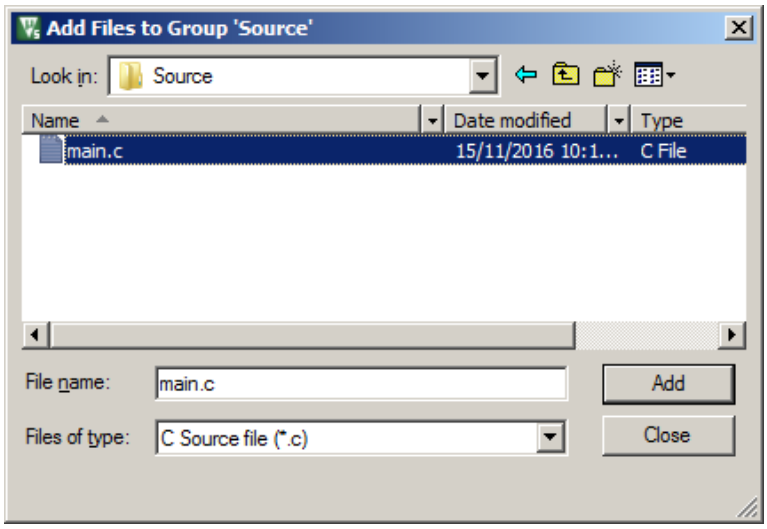
- Chọn file STARTUP.A51 và Add, sau đó Close.



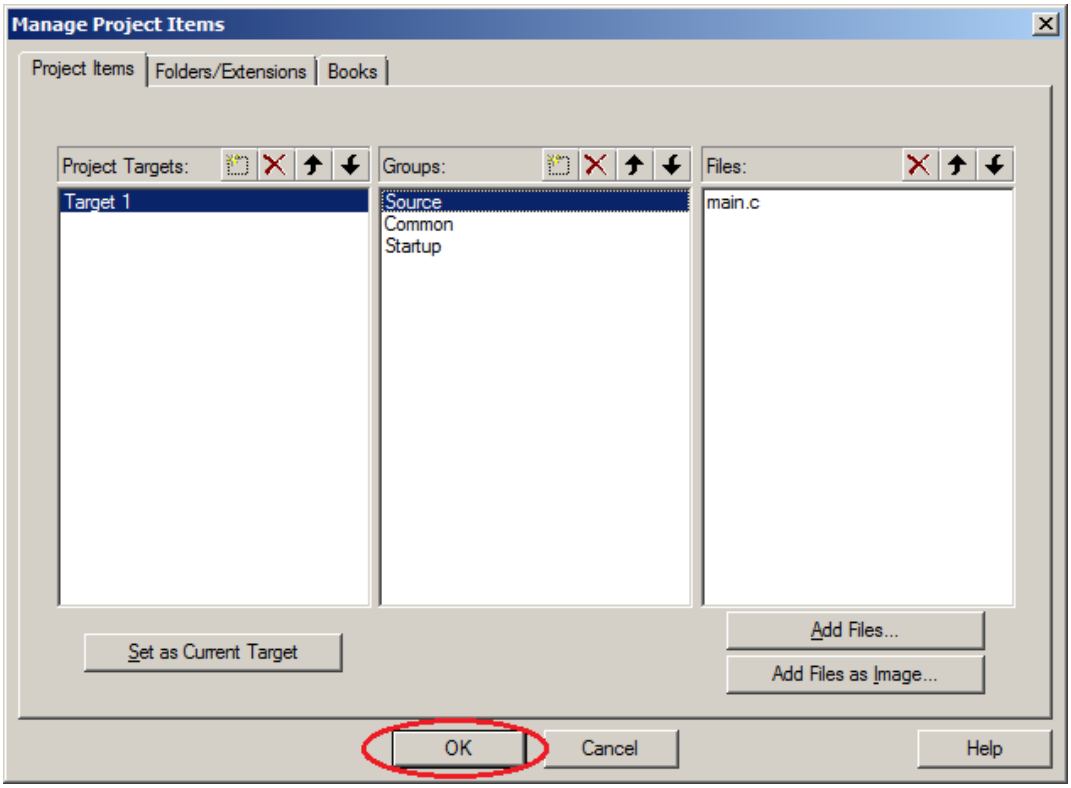
- Làm tương tự với group common, đường dẫn là …/N78E055A/Common



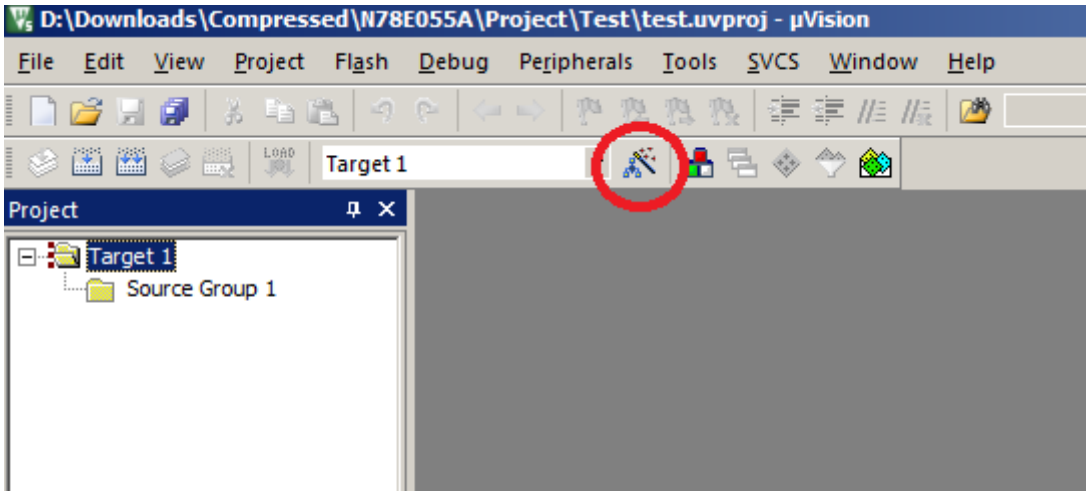
- Với group Source là nơi chứa các file code do mình tự viết, đường dẫn …/N78E055A/Project/Test/Source



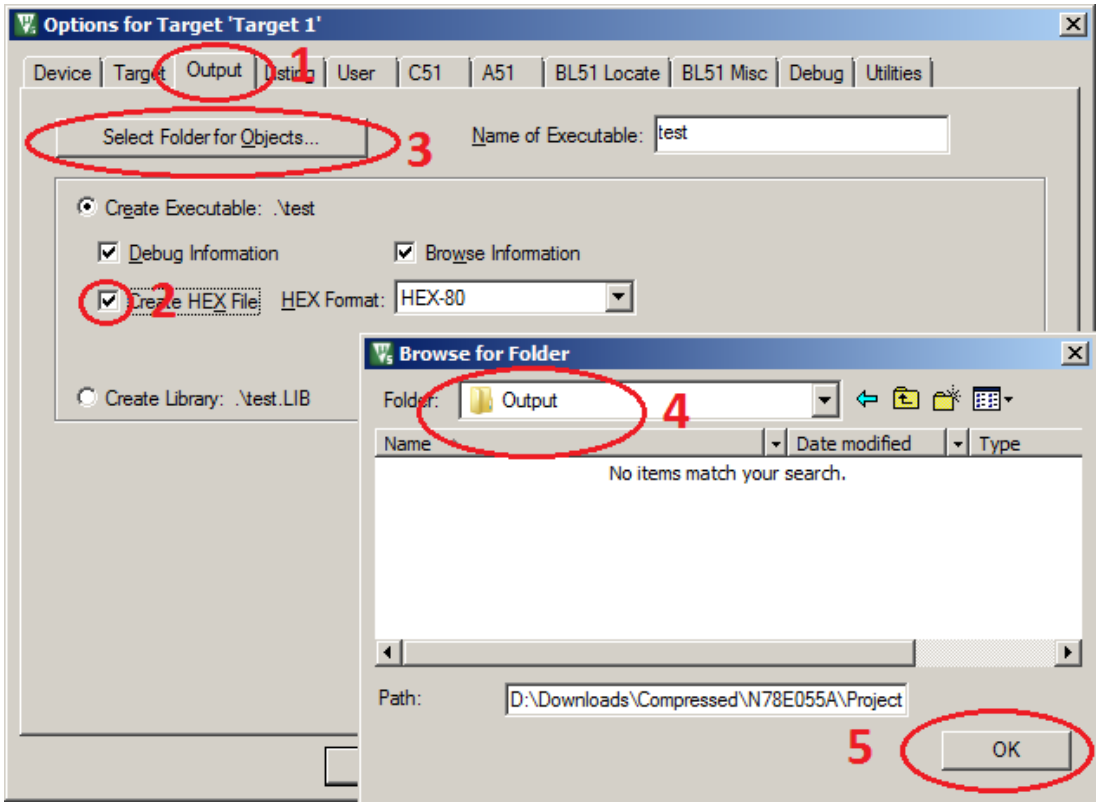
- Sau khi thêm các file nguồn và thư viện ta chọn OK



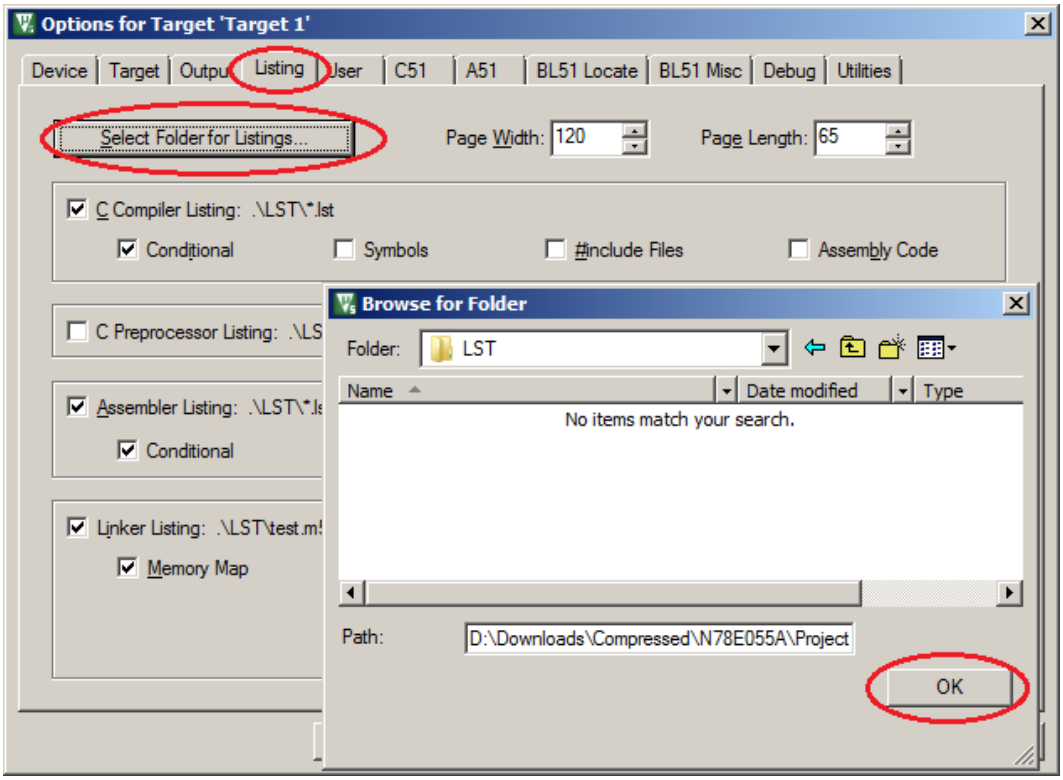
- Chọn Option for target trên thanh công cụ



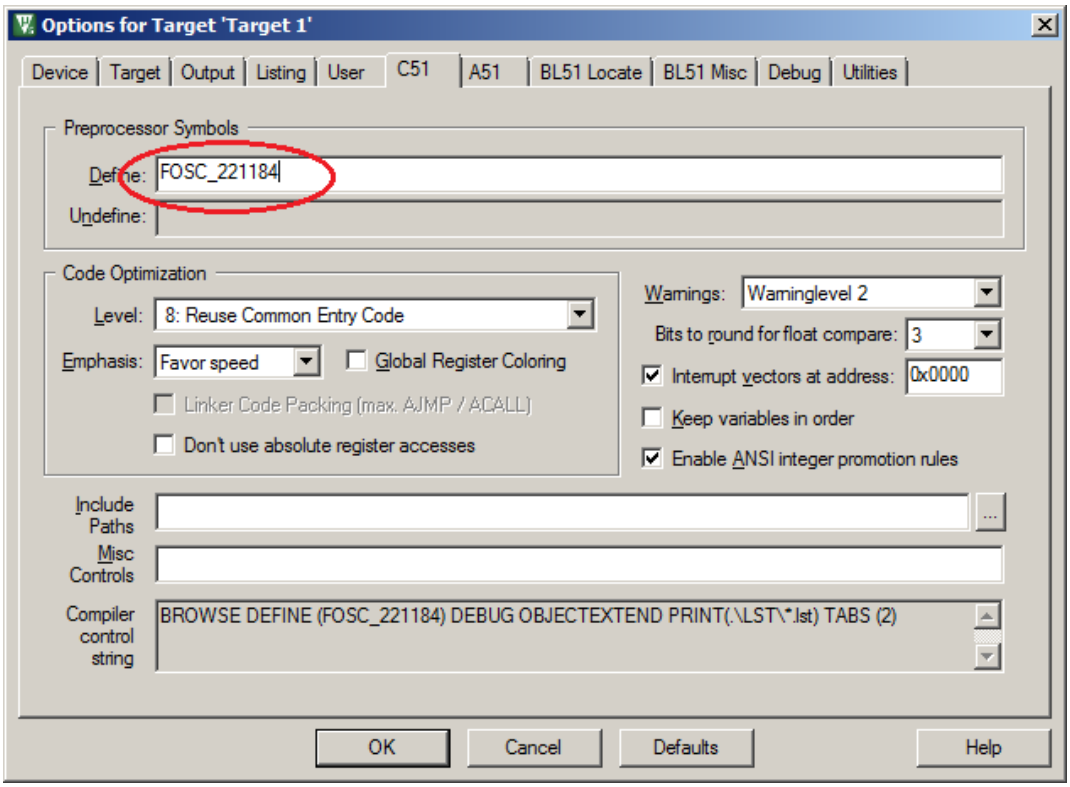
- Trong tab Output, chọn các bước như hình



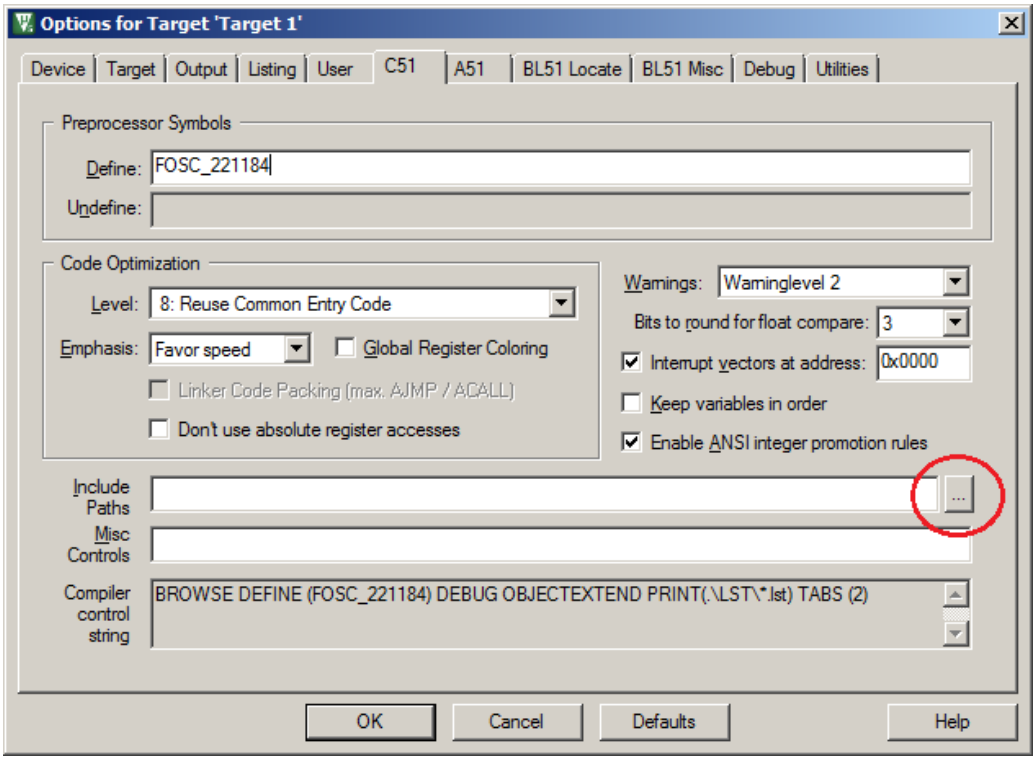
- Trong tab Listing chọn như trong hình, đến thư mục LST trong Test

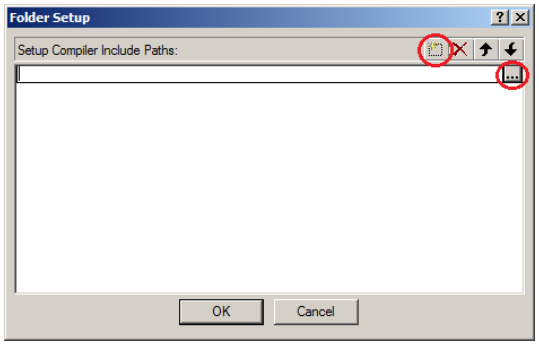


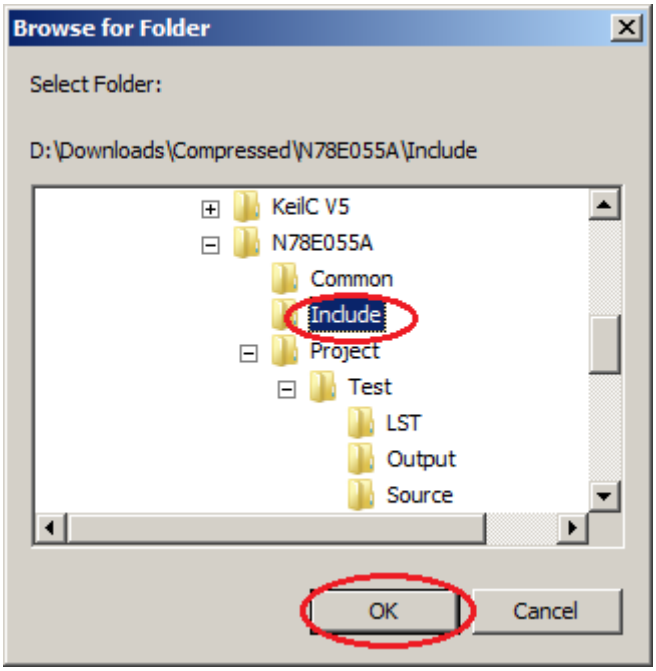
- Dòng define trong tab C51 dùng để khai báo các định nghĩa sử dụng trong project, các định nghĩa ngăn cách bởi dấu “,”. Trước hết ta dùng định nghĩa tần số clock: “FOSC_xxxxxx” với xxxxxx là tần số dao động, có thể xem trong delay.c trong group common



- Include path dùng để thêm các thư viện dùng trong project





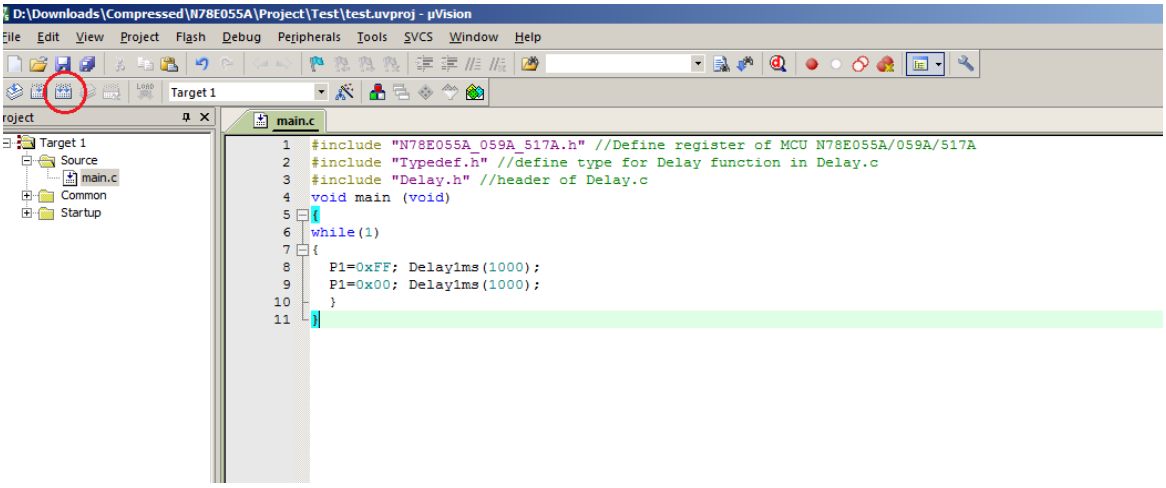


- Sau khi đã thiết đặt các thông số ta chọn OK và bắt đầu viết chương trình.

Sau đây là đoạn code cho chương trình đầu tiên, nháy led trên board AT89S52 v3. Nhìn vào sơ đồ nguyên lý của board kit, ta có thể thấy 7 led đơn được nối với các chân của port 1 vi điều khiển. Coppy đoạn code vào file *main.c* trong group *Source*.

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
void main (void)
{
while(1)
        {
P1=0xFF; Delay1ms(1000);
P1=0x00; Delay1ms(1000);
}
}
```

- Sau khi coppy xong code, ta sẽ combine và build project.

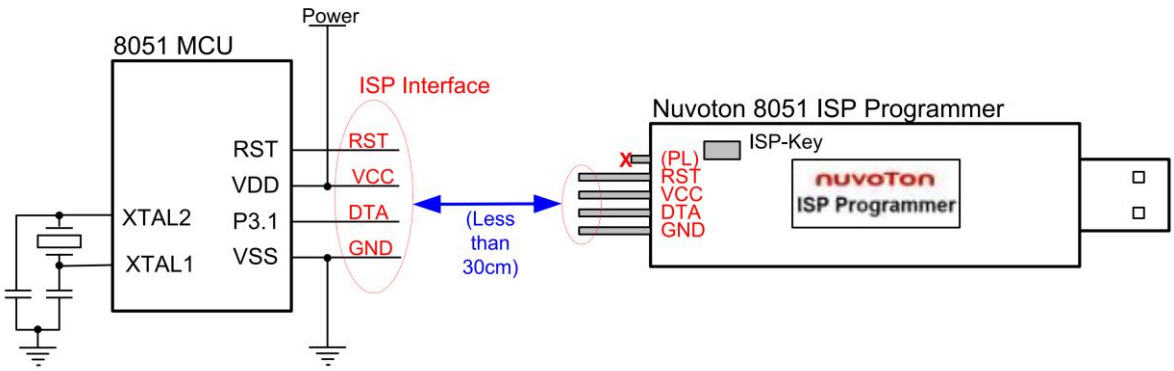


- Sau khi build, ta sẽ có thông báo bên dưới màn hình, không cần chú ý đến warning.

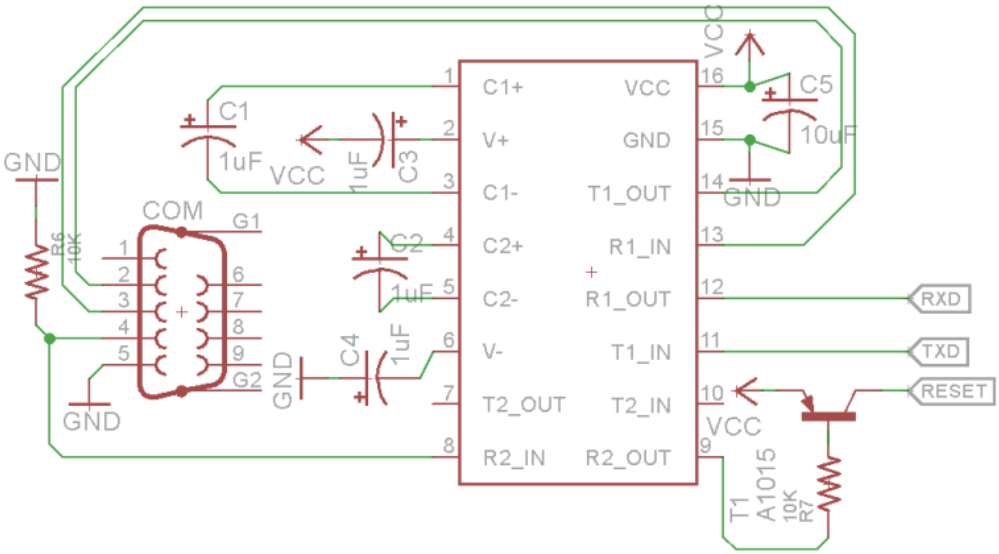
```
Program Size: data=13.0 xdata=0 code=369
creating hex file from ".\Output\test"...
".\Output\test" - 0 Error(s), 1 Warning(s).
```

Bây giờ ta đến bước nạp chương trình vào chip N78E055A. Bởi vì N78E055A hỗ trợ nạp ISP (In-system programmer), và ta dùng mạch nạp ISP-ICP programmer do Nuvoton sản xuất, nên không thể dùng chân nạp của kit AT89S52 mà cần phải đi dây riêng.

Sơ đồ kết nối N78E055A và mạch nạp ISP-ICP của Nuvoton.



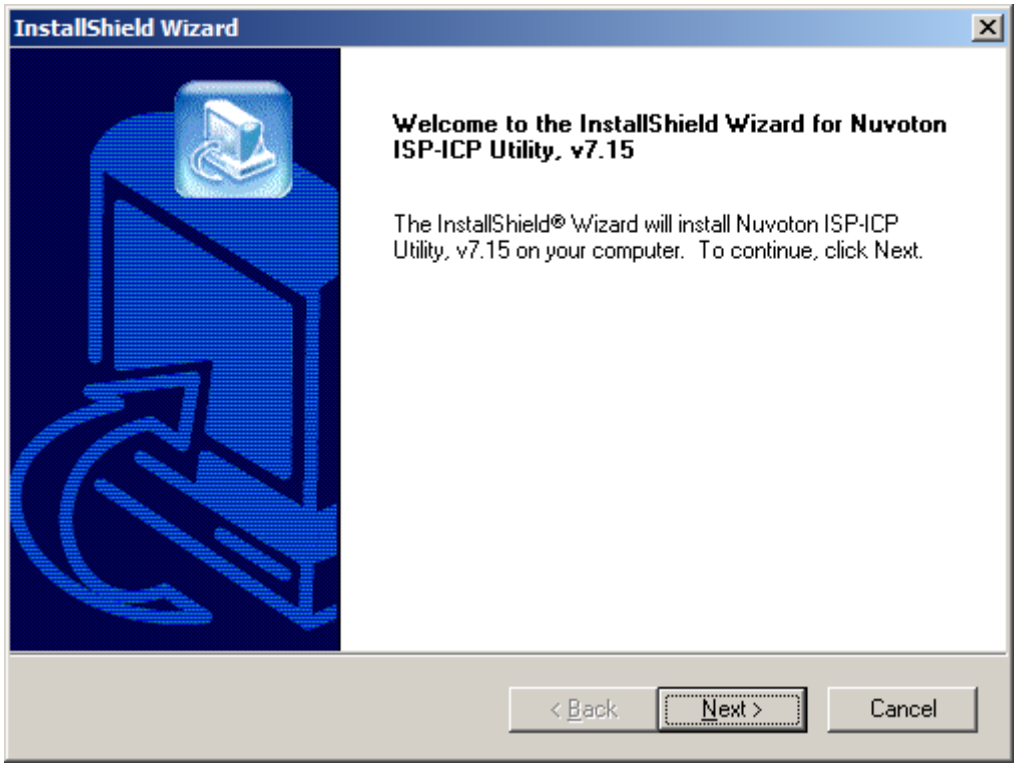
Ngoài ra, N78E055A hỗ trợ nạp ISP qua cổng COM (RS-232), chỉ với Max232 cùng một số linh kiện khác là ta đã có một mạch nạp giá rẻ và tiện lợi, có thể sử dụng mạch PL2303 để kết nối cổng COM ảo UART-USB.



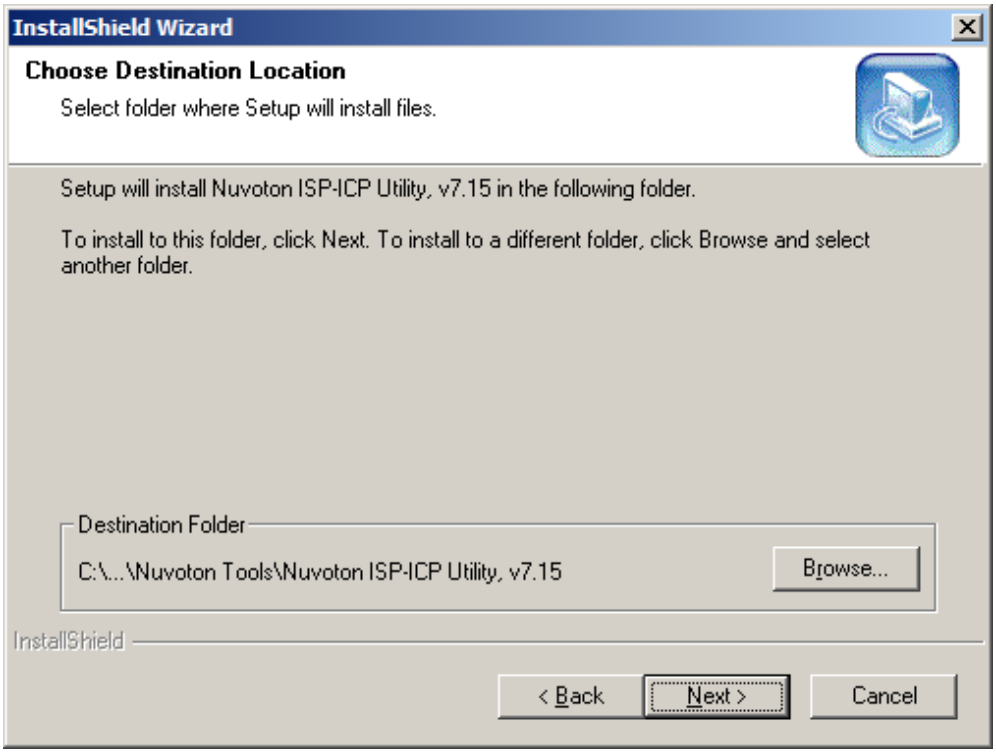
Ta cần chương trình nạp, có thể tải trên trang chủ Nuvoton hoặc phiên bản v7.15 : *https://www.fshare.vn/file/3JAF3TK6VKXS*

Tải xong trình nạp, ta tiến hành giải nén và cài đặt.

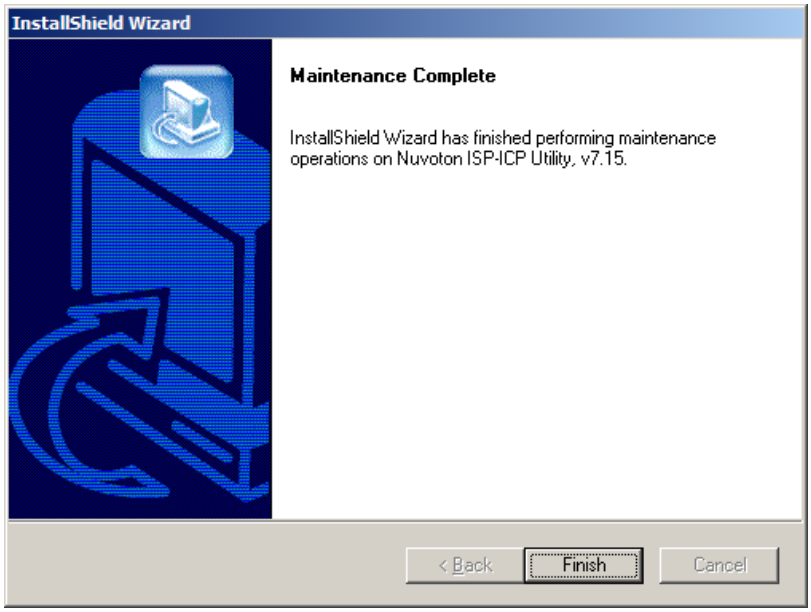
- Chạy file *Setup, ISP-ICP Utility, v7.15.exe* trong thư mục:
  …/(2) Application Program và chọn Next



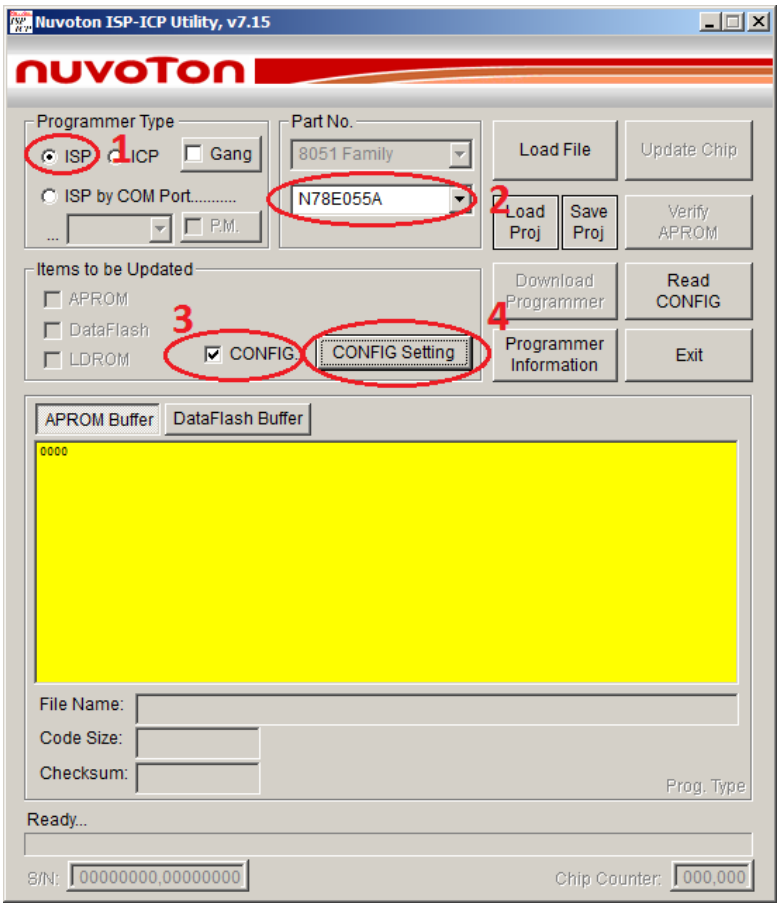
- Chọn đường dẫn đến thư mục cài đặt và Next



- Tiếp tục chọn Next và Finish



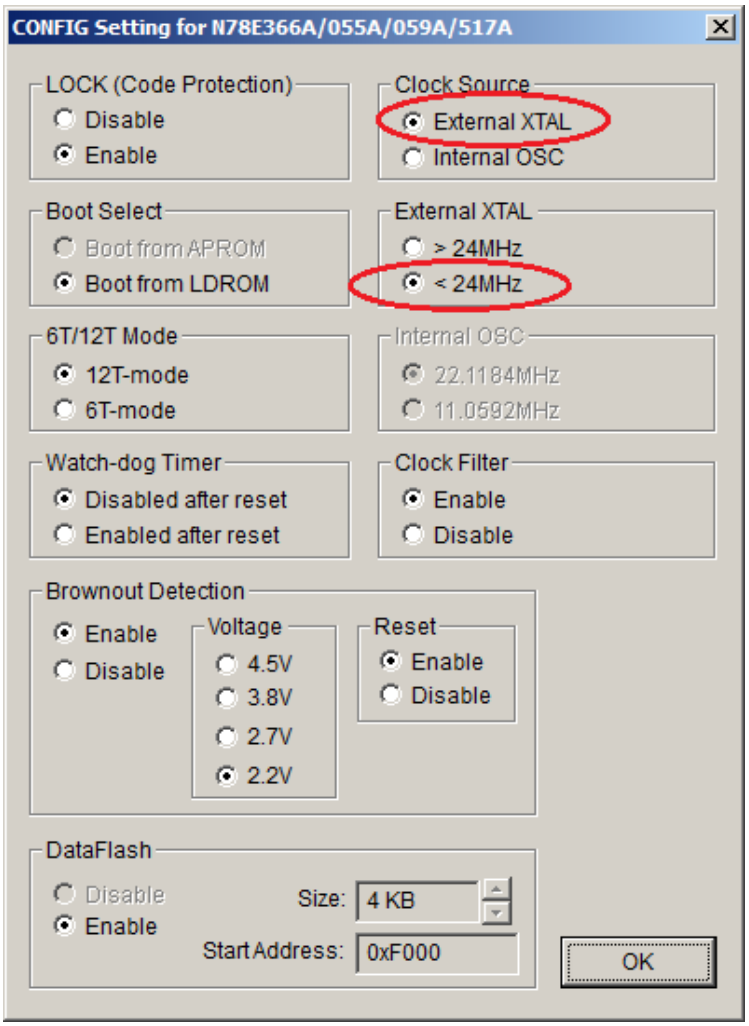
- Chạy chương trình Nuvoton ISP-ICP Utility, và làm theo các bước. Dưới đây là cách nạp theo mạch nạp ISP-ICP programmer của Nuvoton.



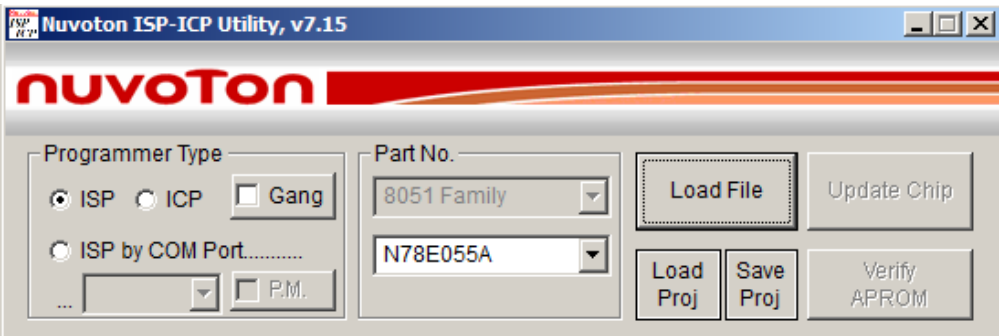
1. *Chọn phương pháp nạp code. N78E055A hỗ trợ nạp code thông qua ISP và ISP qua cổng COM.*
2. *Chọn dòng chip.*
3. *Tích vào để có thể truy cập config*
4. *Config để thiết đặt thông số cho chip.*

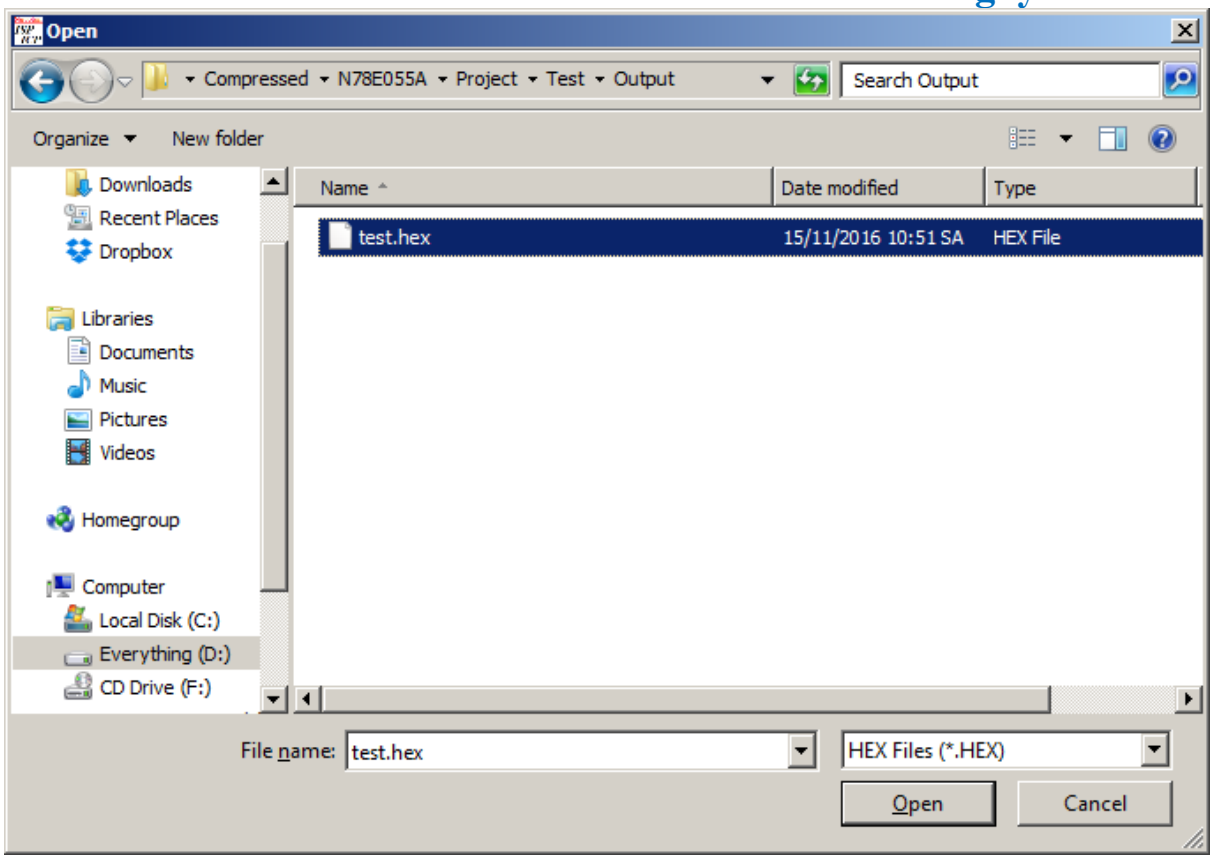
Chú ý: Nếu sử dụng cách nạp thông qua cổng COM thì ta chỉ việc chọn ISP by COM Port….. phần config và nạp code thì tương tự.

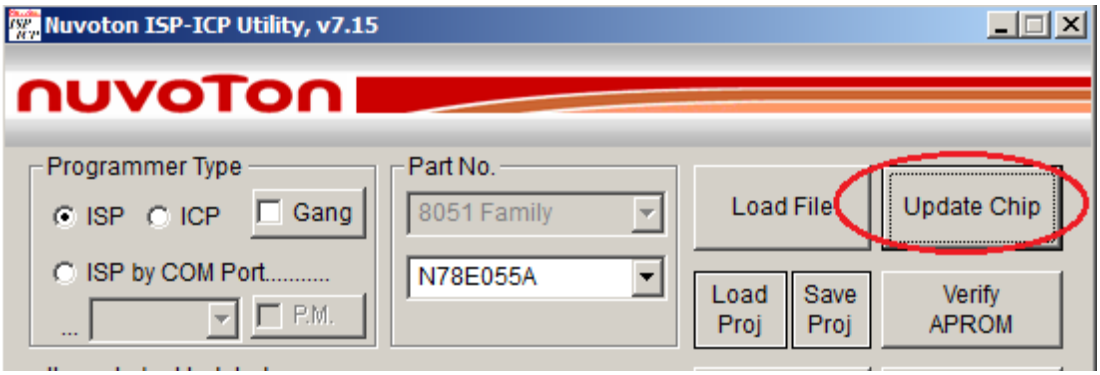
- Phần config chỉ cần thiết đặt cho lần nạp đầu tiên, từ lần sau ta sẽ bỏ tích phần này. Trong các ví dụ tiếp theo, ta sử dụng thạch anh ngoài tần số 22MHz nên chọn như hình.
  Chú ý: nếu như sử dụng thạch anh nội, thì ta chọn Internal OSC, nhưng nạp lần đầu tiên vẫn cần phải có thạch anh ngoài để nạp config, từ lần sau thì không cần nữa.



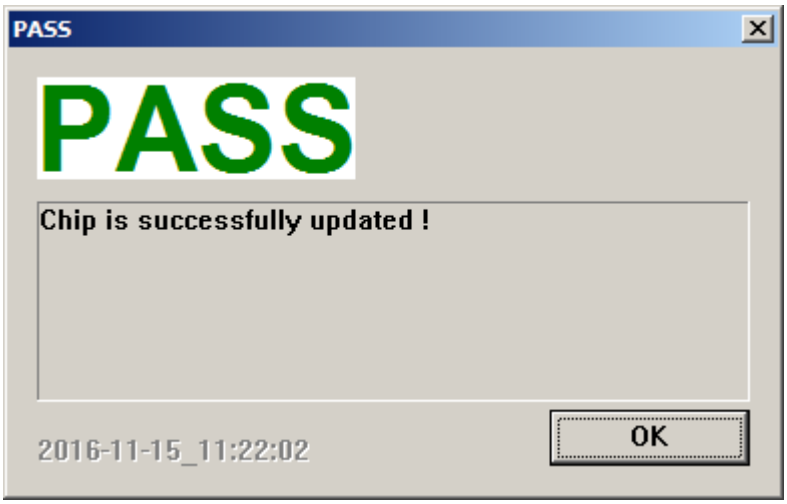
- Ta chọn Load file, dẫn đến thư mục …/Project/Test/Output và chọn file test.hex sau đó chọn Update Chip.







- Sau khi update, ta có thông báo đã nạp xong chương trình cho vđk



Trong N78E055A/N78E059A có 3 bytes thiết lập phần cứng, gọi là CONFIG bytes (như xung hệ thống, chế độ 12T/6T, bảo mật code,..). Những thiết lập có thể thay đổi được thông qua Programmer/Writer hoặc chế độ ISP. Như vậy, ta không thể tác động tới các thiết lập này trong chương trình (APROM) như một số dòng chip khác. Chính vậy, cần chú ý tới tốc độ hệ thống khi giao tiếp UART, Timer, hàm Delay,… và khi đổ code cho MCU bằng phần mềm ‘***’Nuvoton ISP-ICP Utility’’*** cùng với việc thiết lập cho chính xác.

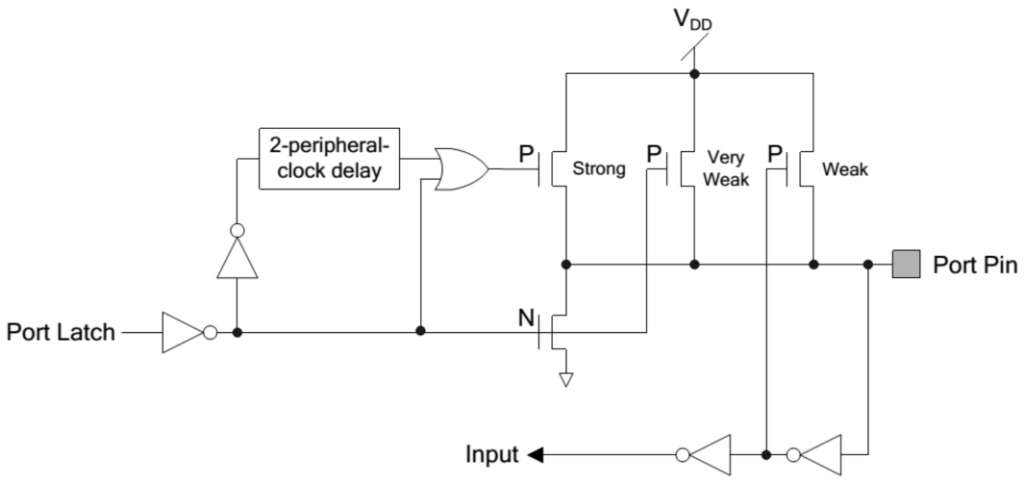
Mặc định ngay từ khi xuất xưởng, MCU được thiết lập :

- Chạy 12T (1 lệnh thực thi cần 12 xung dao động).
- Chạy với bộ giao động ngoài. Chính vậy, trong hướng dẫn này sẽ đổ code lần đầu tiên với thạch anh ngoài rồi thiết lập chạy với IRC để từ lần sau không cần thạch anh nữa.
- Vô hiệu hóa Watch-dog Timer
- Hạn chế lệnh MOVC (lệnh này có thể giúp chương trình bên ngoài MCU đọc dữ liệu trong APROM, LDROM của MCU thông qua giao tiếp mở rộng bộ nhớ của kiến trúc 8051 – hiểu nôm na là : vô hiệu hóa MOVC thì dữ liệu trong APROM và LDROM không bị “đọc trộm” -> bảo mật dữ liệu).
- Mở chế độ lọc xung dao động (Clock filter). Tăng khả năng chống nhiễu, chống ổn cho MCU. Có thể do cảm ứng điện từ hoặc từ board mạch dễ hấp thu dao động ngoài (nhiễu) làm sai lệnh xung hệ thống -> MCU hoạt động không ổn định, chập chờn, chạy lỗi,…Và bộ lọc này sẽ lọc những dao động ngoài mong muốn, thường là các xung gai tần số cao trên 24MHz. Chính vậy, khi sử dụng thạch anh hoặc bộ dao động ngoài từ 24MHz trở lên thì phải vô hiệu hóa bộ lọc này đi.

V. MỘT SỐ PROJECT ĐƠN GIẢN CỦA N78E055A TRÊN KIT AT89S52 V3

NGOẠI VI GPIO CỦA N78E055A

N78E059A/N78E055A có tối đa 5 port-8 bit (mỗi port có 8 chân ra), tùy vào kiểu đóng gói mà đưa chân ra có đầy đủ các port hay không. Được đánh dấu P0 ~ P4, có thể thiết lập cho từng chân đóng vai trò là ngõ ra hoặc ngõ vào hoặc dạng 2 chiều. Với kiểu ngõ ra: với mức logic high/ low, high cho dòng ra (source), low cho dòng vào (sink). Và cũng như hầu hết MCU, N78E055A cho dòng sink lớn hơn là dòng source. Vì vậy, trong các ứng dụng điều khiển, ta vẫn hay cho MCU đóng vai trò kích xuống mass, dùng điện trở ngoài kéo lên (pull-up).

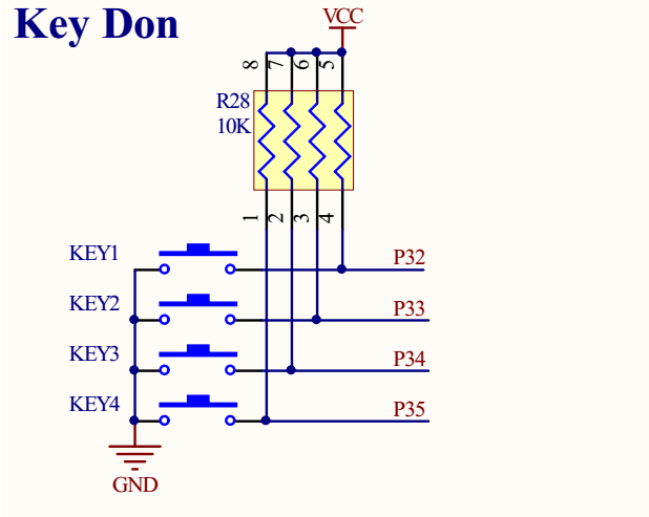


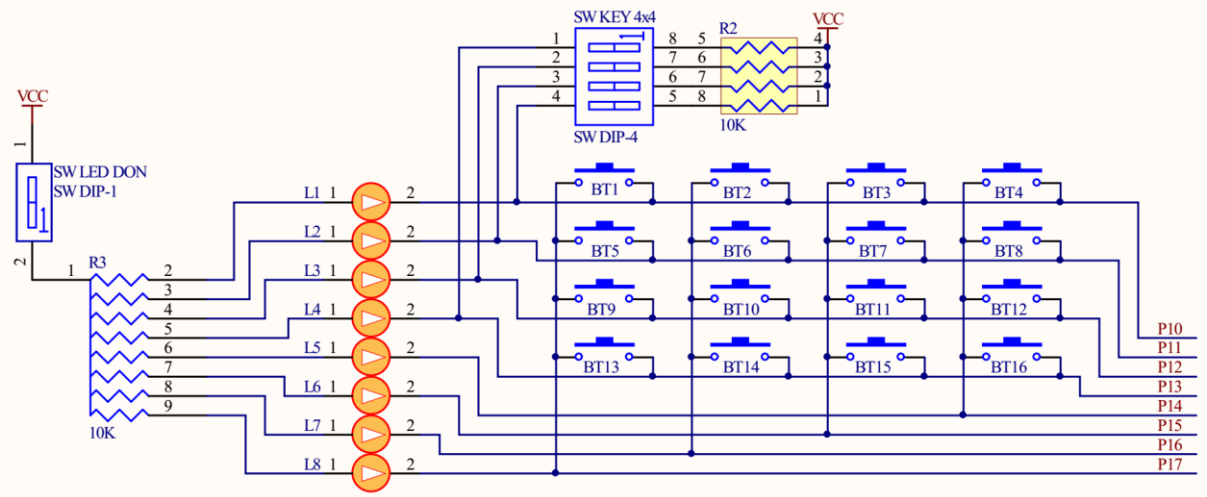
Chúng ta quan tâm đến các thanh ghi về IO của N78E055A như Px (x=1,2,3,4) để điều khiển cả PORT, và Pxy (y=0~7) để truy cập đến từng chân riêng của mỗi Portx.

Ngoài ra với port 0, ta sử dụng P0OR để cho phép sử dụng trở treo nội hay không (0: không cho phép, 1: cho phép) khi sử dụng port 0 là cổng vào input.

1. **PROJECT 01: Điều khiển Led.**

Board AT89S52 thiết kế sử dụng 7 led đơn kết nối với port 1 vi điều khiển, có sử dụng 4 nút bấm đơn tại các chân P3.2, P3.3, P3.4, P3.5. Dựa vào đó, ta viết chương trình điều khiển 7 led đơn bằng các nút bấm trên.



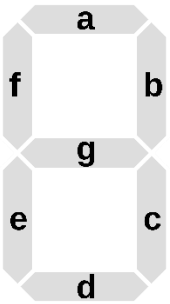


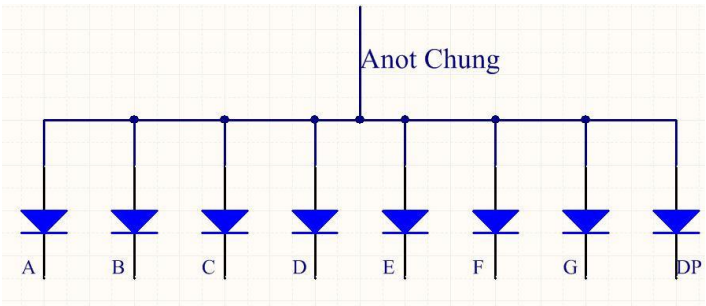
Code: Ta tạo một project mới như những bước tạo project đã nêu trên. Sử dụng đoạn code trong file main.c

```
#include "N78E055A_059A_517A.h" // thu vien thanh ghi cho N78E055A
#include "Typedef.h" //dinh dang kieu bien
#include "Delay.h"   // thu vien ham delay dung timer0
 void main (void){
while(1)
   {
        if(P32==0)   //so sanh muc cua chan 2 cong 3
        {
        while (!P32); // doi den khi tha nut bam
        P1=~P1;
        }
    }
}
```

2. PROJECT 02: Điều khiển led 7 đoạn.

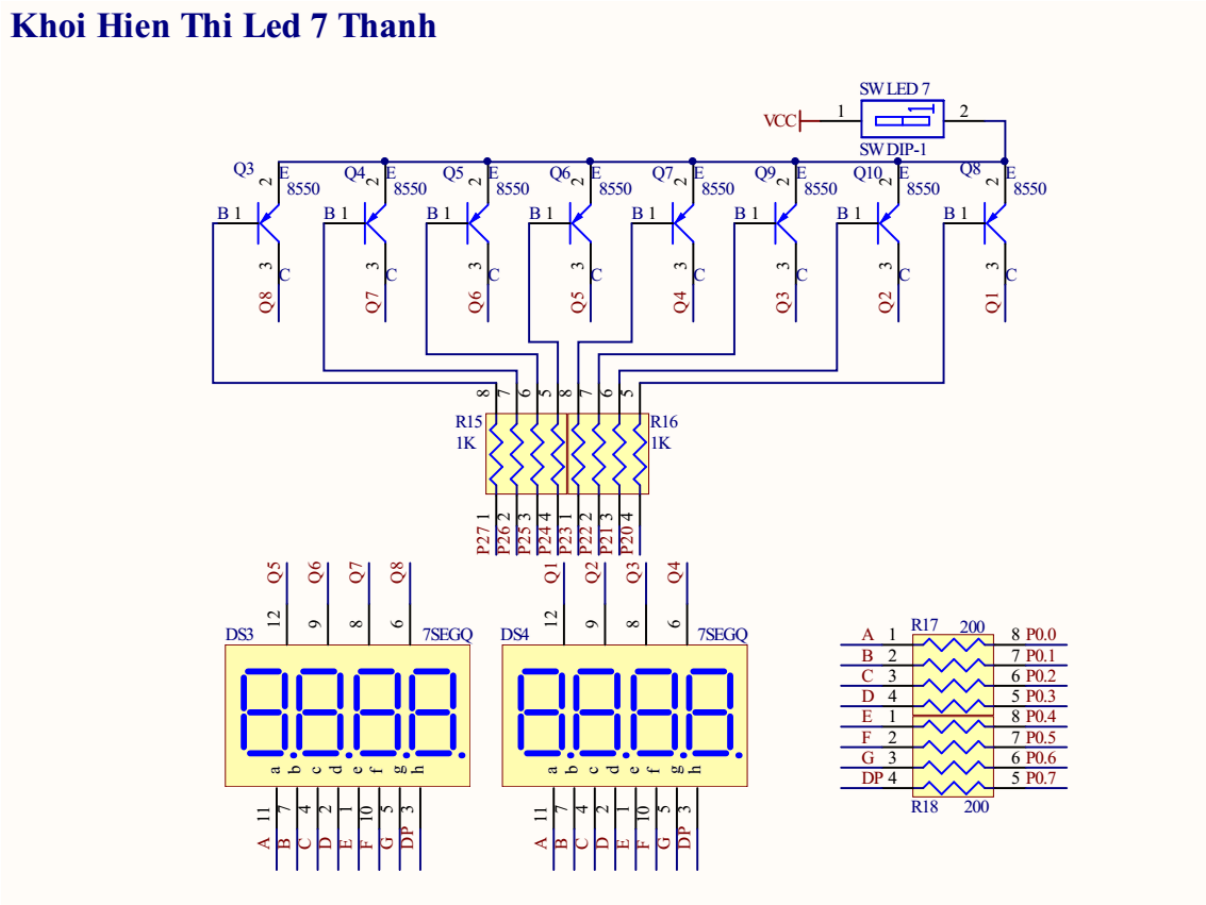
LED 7 đoạn hay LED 7 thanh (Seven Segment display) là 1 linh kiện rất phổ dụng, được dùng như là 1 công cụ hiển thị đơn giản nhất. Trong LED 7 thanh bao gồm ít nhất là 7 con LED mắc lại với nhau, vì vậy mà có tên là LED 7 đoạn là vậy. 7 LED đơn được mắc sao cho nó có thể hiển thị được các số từ 0 - 9 , và 1 vài chữ cái thông dụng, để phân cách thì người ta còn dùng thêm 1 led đơn để hiển thị dấu chấm (dot).





Để hiển thị một số nào đó, ta phải làm cho các led tương ứng sáng hoặc tối. Ví dụ như muốn hiển thị số 4 thì các led b,c,f,g phải sáng, còn các led a,e,d thì tối.

Board Kit sử dụng 2 thanh led 7 đoạn 4 số anot chung điều khiển bằng 2 port P0 và P2, có thể hiển thị số có 8 chữ số. Ý tưởng ở đây là sử dụng nút bấm đơn để tăng giảm số đếm hiển thị trên 2 thanh led 7 đoạn.



Để có thể làm và hiển thị 2 thanh led 7 đoạn 4 số này, ta chỉ cần viết một chương trình sáng lần lượt các led 7 thanh trong một khoảng thời gian rất ngắn. Khi đó ta sẽ nhìn thấy được nội dung hiển thị một cách liên tục.

Code:

```
#include "N78E055A_059A_517A.h"
#include "typedef.h"
#include "Delay.h"
//code display for 7SEG Anode Common 0 1 2 3 4 5 6 7 8 9 A B C D E F
unsigned char data7segAC[17] = {0xC0,0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82,
0xF8, 0x80,
0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E, 0xFF}; //OFF 7SEG};
//-------------------------------------------------------------
unsigned char buffer_led[8];
//-------------dinh nghia cong dieu khien--------------------------------
#ifndef PORT_ADDR
#define PORT_ADDR       P2
```

```
#endif

#ifndef PORT_DATA
#define PORT_DATA       P0
#endif
void Change_AC(unsigned long int number)
{
        unsigned char i,j,sxt;
        unsigned long int temp;
        for(i=7;i>0;i--)
        {
                temp=1;
                for(j=i;j>0;j--)
                {
                        temp*=10;
                }
                buffer_led[i]=data7segAC[number/temp];
                number=number%temp;
        }
        buffer_led[0]=data7segAC[number];
        for(i=0;i<4;i++)
        {
        sxt=buffer_led[i];
        buffer_led[i]=buffer_led[i+4];
        buffer_led[i+4]=sxt;
        }
}
void One_Led_display_AC(unsigned char addr)
{
    PORT_DATA=0xff;
    PORT_ADDR=~(0x01<<addr);
    PORT_DATA=buffer_led[7-addr];
}
void LED7_Display(unsigned long int number)
{
        unsigned char i=0;
        Change_AC(number);
        while(i!=8)
        {
                One_Led_display_AC(++i);
                Delay10us(5);
        }
}
//-------------------------------------------------------------
```

```
unsigned long int dem;
void main (void)
{
while (1)
{
if(P32==0)
        {
        while(!P32);
        dem++;
        }
if(P33==0)
        {
        while(!P33);
        if(dem<=0) dem=0;
        else    dem--;
        }
LED7_Display(dem);
}
}
```

3. PROJECT 03: Hiển thị trên LCD 16x02

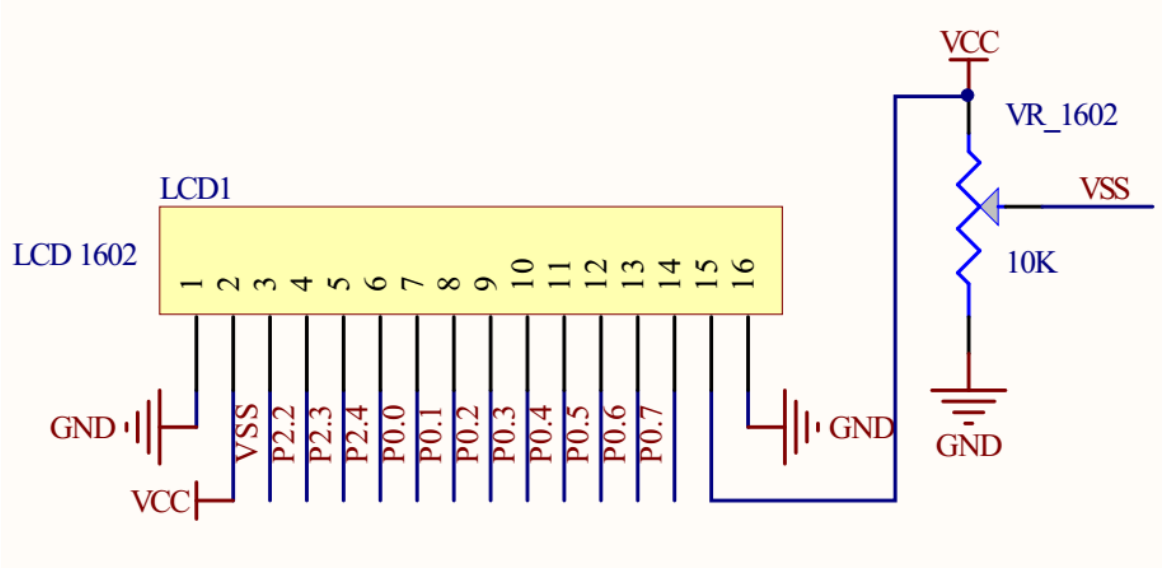
Màn hình LCD 16x02 hay còn gọi màn hình TextLCD dùng để hiển thị chữ cái, ký tự trong bảng mã ASCII, giao tiếp thông qua giao thức nối tiếp (I2C) hoặc song song.

Sơ đồ chân LCD1602:

LCD	Arduino
1 GND	GND
2 VDD	5V
3 Contrast	Resistor to GND
4 RS	PIN 0
5 R/W	GND
6 Enable	PIN 1
7 DB0	Not connected
8 DB1	Not connected

LCD	Arduino
9 DB2	Not connected
10 DB3	Not connected
11 DB4	PIN 2
12 DB5	PIN 3
13 DB6	PIN 4
14 DB7	PIN 5
15 Back LED+	Resistor to 5V
16 Back LED-	GND

Nhìn vào sơ đồ nguyên lý của kit AT89S52, các chân RS, RW, E của lcd 16x2 được kết nối với 3 chân 2.2, 2.3, 2.4 của vi điều khiển, các chân data D0->D7 đc kết nối với port 0, thích hợp cho giao tiếp 8bit và 4bit. Trong project này, ta sẽ sử dụng giao tiếp 4bit. Sử dụng nút bấm để đếm từ 0 đến 9.



Đầu tiên ta tạo 2 file thư viện dành cho lcd 16x02: file *16x2.h* ở thư mục include và *16x2.c* ở thư mục common.

Trong file *16x2.h*:

```
#define LINE_1 0x80
#define LINE_2 0xC0
#define CLEAR_LCD 0x01
//---------khai bao chan dieu khien lcd
#define LCD_RS          P22
#define LCD_RW          P23
#define LCD_E           P24
#define LCD_D4          P04
#define LCD_D5          P05
#define LCD_D6          P06
#define LCD_D7          P07

//-----------khai bao ham dieu khien
void lcd1602_enable(void);
void lcd1602_send_4bit_data ( unsigned char cX );
void lcd1602_send_command (unsigned char cX );
void lcd1602_init (void );
void lcd1602_gotoxy(unsigned char x, unsigned char y);
void lcd1602_clear(void);
void lcd1602_putchar ( unsigned int cX );
void lcd1602_puts (char *s);
```

Trong file *16x2.c*:

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
```

```
#include "16x2.h"
// Ham Khoi Tao LCD
 void lcd1602_enable(void)
{
LCD_E=1;
Delay10us(1);
LCD_E=0;
Delay10us(1);
}
// ---------------------------------------------------
// Ham Gui 4 Bit Du Lieu Ra LCD
 void lcd1602_send_4bit_data ( unsigned char cX )
{
 LCD_D4 = cX & 0x01;
 LCD_D5 = (cX>>1)&1;
 LCD_D6 = (cX>>2)&1;
 LCD_D7 = (cX>>3)&1;
}
// ---------------------------------------------------
// Ham Gui 1 Lenh Cho LCD
 void lcd1602_send_command (unsigned char cX )
{
lcd1602_send_4bit_data  ( cX >>4 );      // send 4 bit high
lcd1602_enable() ;
lcd1602_send_4bit_data ( cX  );                                  // send 4 bit low
lcd1602_enable() ;
}

// ---------------------------------------------------
// Ham Khoi Tao LCD
 void lcd1602_init ( void )
{
lcd1602_send_4bit_data ( 0x00 );
Delay1ms(200);
LCD_RS=0;
LCD_RW=0;
LCD_E=0;
lcd1602_send_4bit_data ( 0x03 );         // ket noi 8 bit
lcd1602_enable() ;
lcd1602_enable () ;
lcd1602_enable () ;
lcd1602_send_4bit_data ( 0x02 );                         // ket noi 4 bit
lcd1602_enable() ;
```

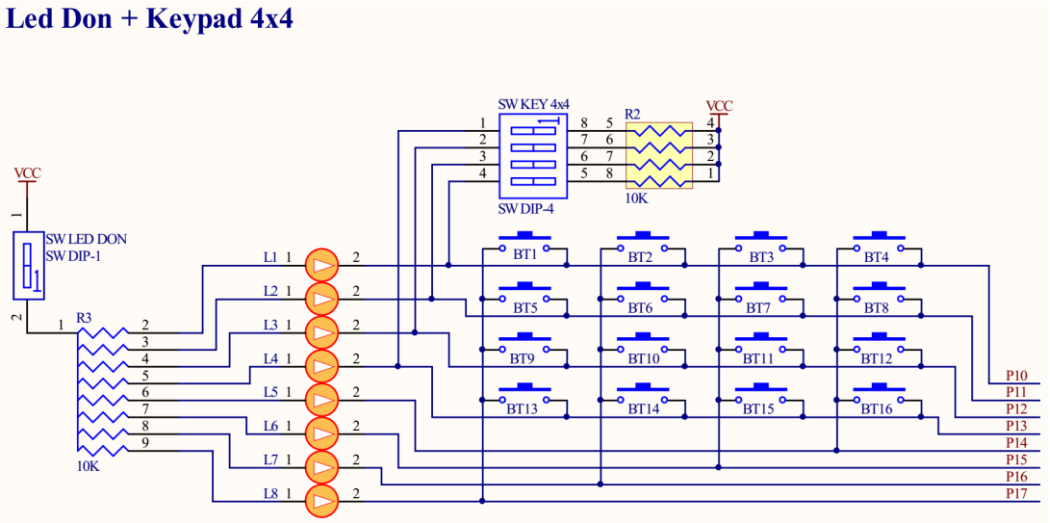
```
lcd1602_send_command( 0x2C );
lcd1602_send_command( 0x80);
lcd1602_send_command( 0x0C);
lcd1602_send_command( 0x06 );
lcd1602_send_command( CLEAR_LCD );        }

// ------------------------------------------------
// Ham Thiet Lap Vi Tri Con Tro
void lcd1602_gotoxy(unsigned char x, unsigned char y)
{
  unsigned char address;
  if(!y)
  address = (LINE_1+x);
  else
  address = (LINE_2+x);
  Delay1ms(3);
  lcd1602_send_command(address);
  Delay1ms(1);
}
// ------------------------------------------------
// Ham Xoa Man Hinh LCD
void lcd1602_clear(void)
{
lcd1602_send_command( CLEAR_LCD );
  //delay_us(300);
}
// ------------------------------------------------
// Ham Gui 1 Ky Tu Len LCD
 void lcd1602_putchar ( unsigned int cX )
{
LCD_RS=1;
lcd1602_send_command( cX );
  LCD_RS=0;
}
// ------------------------------------------------
// Ham Gui 1 Chuoi Ky Tu Len LCD
void lcd1602_puts(char *s)
{
   while (*s)
   {
                    lcd1602_putchar(*s);
      s++;
   }
}
```

Trong chương trình chính main.c:

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
#include "16x2.h"
//-----------------------------
unsigned char dem;
void main (void)
{
lcd1602_init();
lcd1602_clear();
lcd1602_gotoxy(0,0);
lcd1602_puts("Testing....");
Delay1ms(500);
while(1)
   {
        if(P32==0) {while(!P32); if (dem>=9) dem=9; else dem++;}
        if(P33==0) {while(!P33); if (dem<=0) dem=0; else dem--;}
        lcd1602_gotoxy(0,1);
        lcd1602_putchar(dem+48);
   }
}
```

4. **PROJECT 04: Ma trận phím 4x4**

Ma trận phím 4x4 có 16 nút bấm được sắp xếp theo 4 hàng 4 cột. Các nút bấm trong cùng 1 hàng, 1 cột được nối với nhau nên ta sẽ có 8 đầu ra. Như trong kit AT89S52 v3, có sử dụng 4 chân P1.0, P1.1, P1.2, P1.3 là input, 4 chân P1.4, P1.5, P1.6, P1.7 là output (mức thấp).



Sau đây là đoạn code chương trình giao tiếp với ma trận phím 4x4:

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
```

```
#include "16x2.h"

unsigned char matrix_pad[4][4]={
{'A','3','2','1'},
{'B','6','5','4'},
{'C','9','8','7'},
{'D','*','0','#'}
};

unsigned char scan_matrix()
{
    unsigned char r,c;
    P1=0x0f;
    Delay10us(1);
    if((P1&0x0f)!=0x0f)

    {
            Delay10us(10);
            if((P1&0x0f)!=0x0f)
            {
                    for(c=0;c<4;c++)
                    {
                            P1=~(0x01<<(4+c));
                            Delay1ms(1);
                            for(r=0;r<4;r++)
                            {
                                    if((P1&(0x01<<r))==0)
                                    {
                                    while((P1&0x0f)!=0x0f);
                                    return matrix_pad[r][c];
                                    }
                            }
                    }
            }
}
    return 0;
                            }

unsigned char pad1,pad,i;
void main(void)
{
    lcd1602_init();
    lcd1602_gotoxy(0,0);
    lcd1602_puts("Test matrix 4x4");
```

```
        i=0;
        while(1)
        {
                pad=scan_matrix();
                if(pad!=0)
                {
                        lcd1602_gotoxy(i,1);
                        lcd1602_putchar(pad);
                        i++;
                        if(i>=16)
                        {
                                lcd1602_gotoxy(0,1);
                                lcd1602_puts("                ");
                                i=0;
                        }
                }
        }
    }
```

Ngoại vi Interrupt

Trong N78E055A có 11 nguồn ngắt và có 4 mức ưu tiên. 11 nguồn ngắt từ các ngoại vi: Timer1/2/3/, ngắt ngoài INT0/1/2/3, ngắt UART, SPI, Brown-out, Waking-up Timer (thức dậy từ chế độ Power Down). Mỗi nguồn ngắt đều có các thanh ghi thiết lập cho phép ngắt, mức ưu tiên, cờ báo ngắt, địa chỉ vector ngắt. N78E055A cũng như các MCU 8051 có 4 mức ưu tiên ngắt (level 0 -> level 3), giá trị cao hơn thì ưu tiên cao hơn.

Interrupt Priority Control Bits		Interrupt Priority Level
IPH / EIPH	IP / EIP / XICON[7,3]	
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

Địa chỉ vector ngắt của mỗi nguồn ngắt ta xem trong datasheet để MCU truy cập đúng vị trí. Khi ngắt xảy ra, MCU sẽ nhảy tới địa chỉ này để thực thi chương trình trong ngắt.

Source	Vector Address	Vector Number	Source	Vector Address	Vector Number
External Interrupt 0	0003H	0	Timer 0 Overflow	000BH	1
External Interrupt 1	0013H	2	Timer 1 Overflow	001BH	3
Serial Port Interrupt	0023H	4	Timer 2 Overflow/Capture/Reload	002BH	5
External Interrupt 2	0033H	6	External Interrupt 3	003BH	7
SPI Interrupt	0043H	8	Power Down Waking-up Timer Interrupt	004BH	9
Brown-out Detection Interrupt	0053H	10			

Các thanh ghi liên quan tới việc ngắt:

- IE, EIE và XICON- Cho phép nguồn nào được ngắt, EIE là thanh ghi mở rộng của IE. XICON thiết lập cho INT2 và INT3.
- IP và IPH – Thiết lập mức ưu tiên cho nguồn ngắt. Mỗi nguồn ngắt có 2 bit thiết lập mức ưu tiên (giá trị: 00 ~ 11). IP chứa bit thấp, IPH chứa bit cao của 2 bit thiếp lập này.
- EIP và EIPH – mở rộng cho IP và IPH.
- TCON – đây là thanh ghi điều khiển Timer0/1 nhưng có 4 bit thiết lập kiểu ngắt, flag của INT1, INT0.
- Ngoài ra, mỗi ngoại vi có những thanh ghi riêng cho mình. Vì dụ: Timer có thiết lập kiểu ngắt mức/ cạnh,…

Tùy vào thiết lập và nguồn ngắt mà các cờ ngắt tự xóa bằng phần cứng hoặc ta phải xóa trong chương trình.

5. PROJECT 05: Ngắt ngoài

N78E055A có 2 nguồn ngắt ngoài INT0, INT1 tương ứng với 2 chân P3.2 và P3.3 là 2 nút bấm key 1 và key 2 trong kit AT89S52 v3. Việc ngắt được xác định theo 2 trigger: theo mức thấp (low level) và theo sườn xuống (falling edge).

Ta sẽ sử dụng màn hình lcd16x2 để hiển thị số đếm theo 2 ngắt trên với các trigger khác nhau.

Trong file main.c:

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
#include "16x2.h"

unsigned int dem;
//////////////////////INT0 interrupt subroutine
void INT0_ISR (void) interrupt 0
{
EX0 = 0; // vo hieu hoa ngat INT0
Delay1ms(100);
dem++;
EX0 = 1;      // cho phep ngat INT0
}
//////////////////////INT1 interrupt subroutine
void INT1_ISR (void) interrupt 2
{
EX1 = 0; // vo hieu hoa INT1
Delay1ms(100);
dem--;
EX1 = 1;      // cho phep ngat
}
void lcd_puts_number(unsigned int num) //ham hien thi so co 4 chu so
{
unsigned char n,t,c,dv;
n=num/1000;
t=(num%1000)/100;
```

```
c=(num%100)/10;
dv=num%10;
lcd1602_putchar(n+48);
lcd1602_putchar(t+48);
lcd1602_putchar(c+48);
lcd1602_putchar(dv+48);
}

void main(void)
{
lcd1602_init();
lcd1602_gotoxy(0,0);
lcd1602_puts("Testing.....");
IT0 = 0; //ngat int0 khi o muc 0
IT1 = 1; //ngat theo suon xuong
IPH = 0x04; // muc uu tien cua ngat
PX0 = 1; //bit 0 thanh ghi IP, ket hop IPH de thiet dat muc uu tien
PX1 = 0; //bit 2 thanh ghi IP, ket hop IPH de thiet dat muc uu tien
EX0 = 1; //cho phep ngat int0
EX1 = 1; //cho phep ngat int1
EA = 1; // ngat toan cuc
dem=1000;
while(1)
{
lcd1602_gotoxy(0,1);
lcd_puts_number(dem);
}
}
```

6. PROJECT 06: Timer

N78E055A có 3 Timer/Counter 16 bit (xin gọi tắt là TC), mỗi timer có 2 thanh ghi 8bit cho việc đếm: trong đó THx là 8bit cao, TLx là 8 bit thấp, x= 0/1/2.

TCON, TMOD là các thanh ghi thiết lập cho TC 0 và 1.

T2CON, T2MOD, RCAP2L, RCAP2H là các thanh ghi thiết lập cho TC 2.

Cụ thể chức năng từng bit trong các thanh ghi của TC, ta xem trong *10. Timer/Counters* của datasheet.

TC 0 và 1 có 4 chế độ: 13bit timer (Mod 0), 16bit timer (Mod 1), 8 bit timer tự động nạp lại (Mod 2), 2 timer-8bit song hành (Mod3).

TC2 có các chế độ: Capture mode, Timer 16bit tự nạp lại, chế độ bộ phát baud rate cho UART, xuất xung dao động ra ngoài.

Thư viện delay.c sử dụng timer 0 để tính toán thời gian delay một cách tương đối chính xác. Ta sẽ sử dụng timer 0 để tạo một đồng hồ bấm giờ đơn giản, hiển thị lên lcd.

Code:

```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Delay.h"
#include "16x2.h"

void lcd_puts_time(unsigned char num)
{
	unsigned char c,dv;
	c=(num%100)/10;
	dv=num%10;
	lcd1602_putchar(c+48);
	lcd1602_putchar(dv+48);
}
unsigned char s,m,h;
void main(void)
{
	lcd1602_init();
	lcd1602_gotoxy(0,0);
	lcd1602_puts("  Timer 0  ");
	s=0;
	while(1)
	{
		s++;
		if(s==60)
		{
			m++;
			if(m==60)
			{
				h++;
				m=0;
			}
			s=0;
		}
		lcd1602_gotoxy(0,1);
		lcd_puts_time(h);lcd1602_putchar(58);
		lcd_puts_time(m);lcd1602_putchar(58);
		lcd_puts_time(s); Delay1ms(1000);
	}
}
```

7. PROJECT 07: PWM

Phương pháp điều xung PWM là phương pháp điều chỉnh điện áp ra tải theo một chu kỳ nhất định, có đại lượng đặc trưng là tần số và độ rộng xung. Từ việc thay đổi độ rộng xung, ta sẽ thay đổi điện áp ra output.

N78E059A/N78E055A có 5 kênh ngõ ra PWM (PWM0~PWM4) kết nối với các chân (P1.3~P1.7) có độ phân giải 8bit (0~255).

Các thanh ghi liên quan đến PWM:

- PWMCON0, PWMCON1: 2 thanh ghi điều khiển cho phép điều chế PWM theo các kênh từ 0~4.
- PWMP: điều khiển chu kì của xung.
- PWMx (x=0~4): độ rộng xung giá trị từ 0~255.

Với AT89S52 board ta có thể dùng chân P1.3 tương ứng là PWM0 để điều khiển âm lượng của còi chíp.

Sau đây là code ví dụ, sử dụng 2 nút bấm để thay đổi giá trị PWM.

```
#include "N78E055A_059A_517A.H"
#include "typedef.h"
#include "Delay.h"

unsigned char duty;
void main(void)
{
PWMCON0=0x05;  //cho phep su dung PWM0 tai chan P1.3
PWMP=255; //Chu kỳ xung 0-255
PWM0=128; // do rong xung 0-255
while(1)
{
        if(P32==0)
        {
                PWM0--;
        if(PWM0<5)PWM0=5;
        }
        if(P33==0)
        {
                while(!P32);
                PWM0++;
                if(PWM0>255)PWM0=255;
        }
}
}
```

8. PROJECT 08: Giao tiếp UART

UART là bộ truyền dữ liệu không đồng bộ, thường được dùng trong máy tính công nghiệp, truyền thông, vi điều khiển,.. Chúng giao tiếp thông qua 2 dây Rx và Tx, có chung đất. Các thông số đặc trưng cho giao tiếp UART là baudrate, frame, start bit, data, parity bit và stopbit. Khung truyền frame hay được sử dụng là 10bit (start+data+stop).

N78E059A/N78E055A có một cổng nối tiếp song công hỗ trợ 3 chế độ cho giao tiếp UART. Để có thể hiểu sâu hơn về các chế độ này, có thể tìm hiểu trên chương 13 datasheet.

Các thanh ghi liên quan:

- SCON: thiết lập chế độ cho serial port
- PCON: với bit SMOD (PCON.7) thiết lập double baud rate.
- SBUF: bộ đệm dữ liệu.

Ý tưởng ở đây là xây dựng một chương trình giao tiếp với máy tính thông qua giao thức UART bằng cổng COM. Ta sẽ xử dụng chân P3.0, P3.1 trong project này. Khi bấm nút, thông tin sẽ được gửi lên máy tính qua các phần mềm như Hercules, Terminal.

Code demo của chương trình

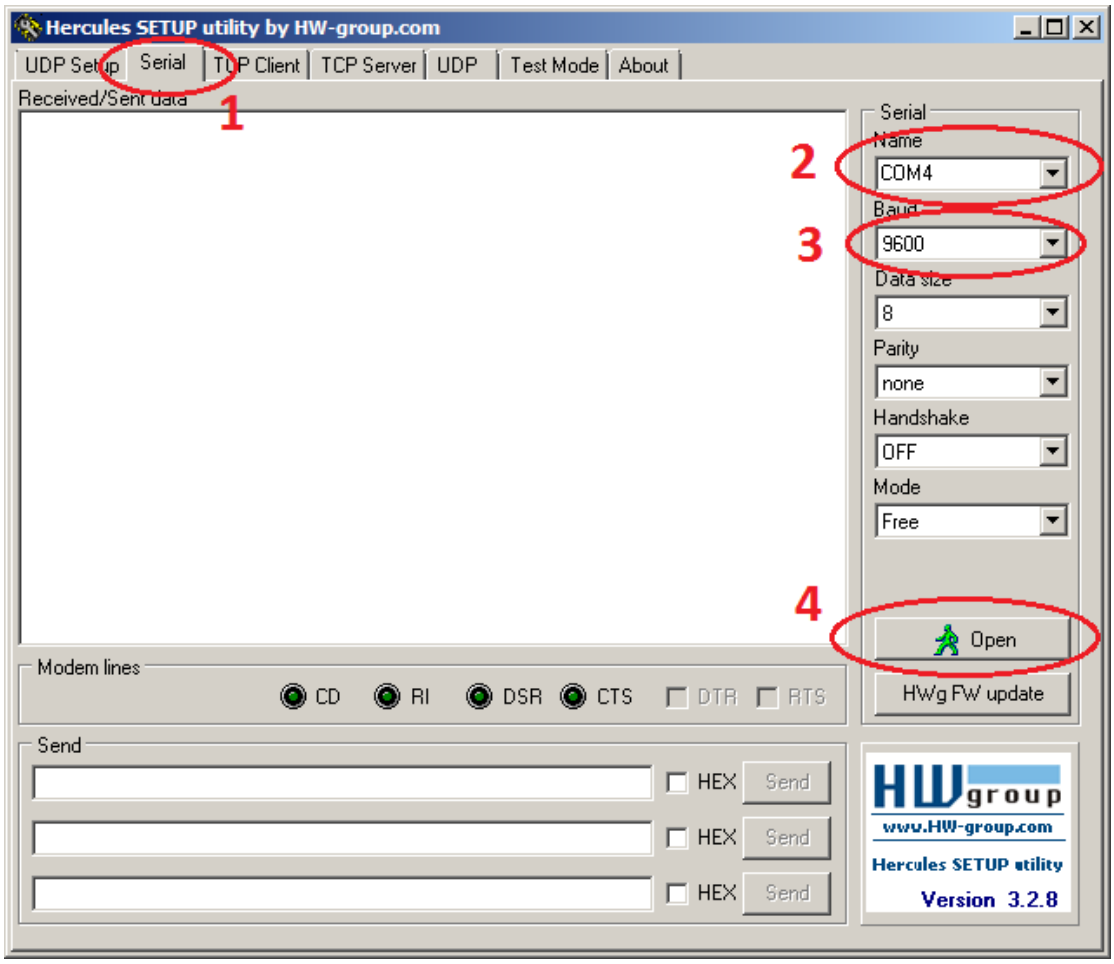
```
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "16x2.h"
#include "Delay.h"
#include <stdio.h> //for "printf"
#define BAUD_RATE 9600
#define U16BAUD_TIMER2 65536-(22118400/BAUD_RATE/32)
char uart_buf_num=0;
char change_flag=0;
char string1[17] = "PC<->UART<->LCD";
char string2[17] = "Click Button!";
void UART_ISR (void) interrupt 4 //ngat uart
{
if (RI) //check Tx or Rx interrupt    // neu co du lieu gui den
{
if(uart_buf_num>15) uart_buf_num=0;
string2[uart_buf_num]= SBUF;
uart_buf_num++;
string2[uart_buf_num]= '\0';
change_flag =2;
RI = 0; //clear RI by software for next reception
}
}
void Timer2_ISR (void) interrupt 5 //interrupt address is 0x002B
{
EXF2 = 0; //Clear Timer 2 External Flag
}
void Timer2_Init (void)
{
TCLK = 1; //select Timer 2 as transmission baud rate source
RCLK = 1; //select Timer 2 as receive baud rate source
```

```
TL2 = RCAP2L = U16BAUD_TIMER2; //baud rate 9600bps@22.1184MHz
TH2 = RCAP2H = U16BAUD_TIMER2 >> 8;
EXEN2 = 1; //enable T2EX pin
ET2 = 1; //Timer 2 interrupt enable, in baud rate generator mode,
//T2EX pin interrupt still works
TR2 = 1; //Timer 2 run
}
void main (void)
{
lcd1602_init(); //Init LCD
lcd1602_gotoxy(0,0);
lcd1602_puts("Test Uart");
Timer2_Init(); //Init timer 2 (source clock of UART)
SCON = 0x52; //initial UART as mode 1, receive enable,
//TI should be set before using "printf"
printf
("\n*==================================================
==========*");
printf ("\n* N78E055A_059A_517A UART Mode 1 Demo *");
printf("\n*==============================================
==============*");
printf ("\n 1. Put character in Terminal software -> display to LCD");
printf ("\n 2. Press BT1, BT2 -> Display to PC (Terminal software)\n");
ES = 1; //enable UART interrupt
EA = 1; //enable global interrupt
uart_buf_num=0;
while(1)
{
if(P32 == 0) //If P32 pin pressed -> increment number
{
while(!P32); //Wait P32 release!
sprintf(string2,"BT1 pressed!");
change_flag=1;
}
if(P33 == 0) //If P32 pressed -> increment number
{
while(!P33); //Wait P32 release!
sprintf(string2,"BT2 pressed!");
change_flag=1;
}
if(change_flag)
{
if(change_flag==1) printf("\n %s", string2);
uart_buf_num=0;
```

```
change_flag=0;
lcd1602_gotoxy(0,1);
lcd1602_puts(string2);
lcd1602_puts("     ");
}
}
}
```

Sau khi đã nạp code cho vi điều khiển, ta sử dụng dây cắm cổng COM để kết nối với máy tính hoặc sử dụng USB UART để kết nối bằng cổng COM ảo.

Tải phần mềm Hercules trên trang chủ *Hw-group* và chạy chương trình.



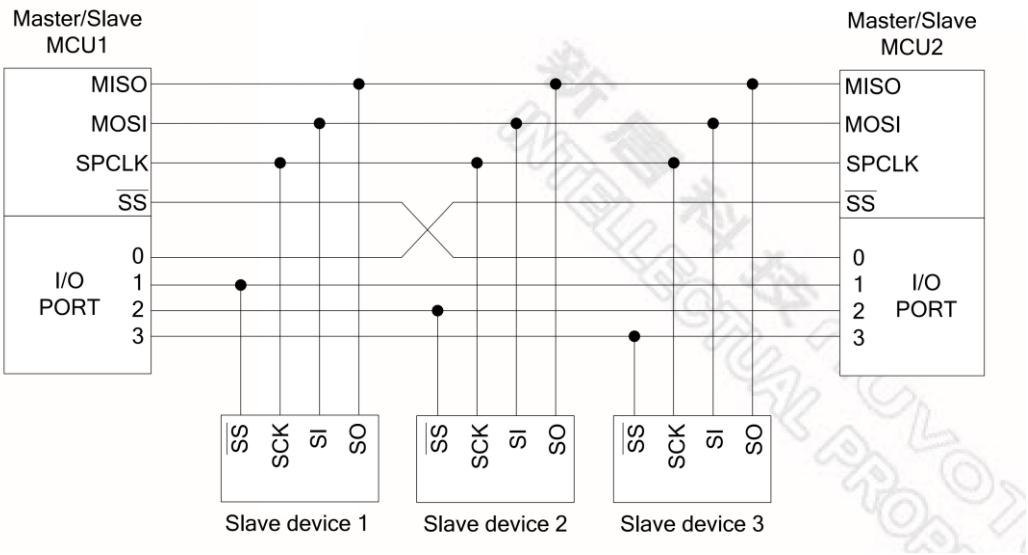
1. Chọn phần giao tiếp nối tiếp Serial.
2. Chọn cổng COM kết nối với máy tính. Nếu sử dụng cổng COM ảo thì có thể vào hardware trong my computer để kiểm tra xem cổng COM là cổng nào.
3. Lựa chọn baundrate phù hợp, 2 thiết bị phải cùng baundrate mới giao tiếp được với nhau.
4. Sau đó ta bật kết nối bằng Open, thử nhấn nút bấm trên bo kit và quan sát kết quả.

9. PROJECT 09: Giao tiếp SPI

Giao tiếp SPI là giao tiếp truyền thông nối tiếp đồng bộ cao theo kiểu Master-Slave. Tức là sẽ có một vi điều khiển, thiết bị đóng vai trò là master sẽ điều khiển, quản lý các

thiết bị slave còn lại. SPI có thể coi như là một phương pháp truyền song công, việc gửi và nhận dữ liệu đồng thời thông qua kết nối 4 dây: CLK, MOSI, MISO, SS. CLK là đầu xung điều khiển của master, cung cấp xung cho các slave. MOSI là chân dữ liệu ra từ master vào slave, MISO ngược lại với MOSI và SS là chân dùng để select slave, lựa chọn slave để truyền thông.

N78E055A hỗ trợ cổng SPI tốc độ cao, với 2 chế độ Master và Slave với tỷ lệ tốc độ cao đến FPERIPH/16 với Master và FPERIPH/4 với Slave. Ngoài ra còn có chế độ chống lỗi, xung đột khi kết nối nhiều Master.



Trong project này ta sẽ sử dụng một con N78E055A ngoài làm master, kết nối với bo kit thông qua chân cắm có sẵn hoặc qua cổng USBISP. Master sẽ gửi dữ liệu theo chu kì nhất định và slave sẽ nhận và hiển thị dữ liệu lên lcd.

Code dành cho master:

```
#include <stdio.h>
#include <intrins.h>
#include <string.h>
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Define.h"
#include "Common.h"
#include "Delay.h"
// định nghia cac bit cua thanh ghi dieu khien giao tiep SPI
#define set_SSOE      SPCR |= SET_BIT7
#define set_SPE       SPCR |= SET_BIT6
#define set_LSBFE     SPCR |= SET_BIT5
#define set_MSTR      SPCR |= SET_BIT4
#define set_CPOL      SPCR |= SET_BIT3
#define set_CPHA      SPCR |= SET_BIT2
#define set_SPR1      SPCR |= SET_BIT1
#define set_SPR0      SPCR |= SET_BIT0
```

```
#define set_DRSS        SPSR |= SET_BIT3
#define set_ESPI        EIE  |= SET_BIT0

#define clr_SSOE        SPCR &= CLR_BIT7
#define clr_SPE         SPCR &= CLR_BIT6
#define clr_LSBFE       SPCR &= CLR_BIT5
#define clr_MSTR        SPCR &= CLR_BIT4
#define clr_CPOL        SPCR &= CLR_BIT3
#define clr_CPHA        SPCR &= CLR_BIT2
#define clr_SPR1        SPCR &= CLR_BIT1
#define clr_SPR0        SPCR &= CLR_BIT0

#define clr_DRSS        SPSR &= CLR_BIT3
#define clr_SPIF        SPSR &= CLR_BIT7

#define SS_PIN          P14

//--------ngat SPI-----------------------------------------------
void SPI_Interrupt_ISR(void) interrupt 8
{
    UINT8 u8I;

    clr_SPIF;
    /* Waiting the Slave ready */
    for(u8I = 0; u8I < 255 ;u8I++);
}
//-------------------------------------------------------------------------------------------
void Enter_Idle_Mode(void)
{
    PCON |= 0x01;
}
//-------------------------------------------------------------------------------------------
void SPI_Error(void)
{
    P2 = 0x78;
    while(1);
}
//----------------Khoi tao SPI-----------------------------------------------
void SPI_Initial(void)
{
    /* Enable Port 0 to weakly pull-up */
    P0OR |= SET_BIT0;
    /* SPI clock = Fosc/128 */
```

```
    set_SPR0;
    set_SPR1;

    /* /SS General purpose I/O ( No Mode Fault ) */
    set_DRSS;
    clr_SSOE;
    /* SPI in Master mode */
    set_MSTR;
    /* MSB first */
    clr_LSBFE;
    clr_CPOL;
    set_CPHA;
    /* Enable SPI function */
    set_SPE;
    /* Enable SPI interrupt */
    set_ESPI;
    EA = 1;
}
//-----------------------------------------------------------------------------------------------
unsigned char i;
void main(void)
{
SPI_Initial();
i=0;
while(1)
{
if(SPI_Send_Byte(CMD_CHECK_BUS)==0)     break;
Delay1ms(500);
}
 while(1)
    {
        i=0;
        for(i=0;i<9;i++)
        {
        SPI_Send_Byte(i+48);
        Delay1ms(1000);
        }
}
}
```

Code dành cho Slave:

```
#include <stdio.h>
#include "N78E055A_059A_517A.h"
#include "Typedef.h"
#include "Define.h"
#include "Delay.h"
```

```
#include "LCD16x2.h"
#define set_SSOE SPCR |= SET_BIT7
#define set_SPE SPCR |= SET_BIT6
#define set_LSBFE SPCR |= SET_BIT5
#define set_MSTR SPCR |= SET_BIT4
#define set_CPOL SPCR |= SET_BIT3
#define set_CPHA SPCR |= SET_BIT2
#define set_SPR1 SPCR |= SET_BIT1
#define set_SPR0 SPCR |= SET_BIT0
#define set_DRSS SPSR |= SET_BIT3
#define set_ESPI EIE |= SET_BIT0
#define clr_SSOE SPCR &= CLR_BIT7
#define clr_SPE SPCR &= CLR_BIT6
#define clr_LSBFE SPCR &= CLR_BIT5
#define clr_MSTR SPCR &= CLR_BIT4
#define clr_CPOL SPCR &= CLR_BIT3
#define clr_CPHA SPCR &= CLR_BIT2
#define clr_SPR1 SPCR &= CLR_BIT1
#define clr_SPR0 SPCR &= CLR_BIT0
#define clr_DRSS SPSR &= CLR_BIT3
#define clr_SPIF SPSR &= CLR_BIT7
#define clr_ESPI EIE |= CLR_BIT0
#define SS_PIN P14

//-----------------------------------------------
void SPI_Interrupt_ISR(void) interrupt 8
{
        clr_SPIF;
        if (SPDR==default)
        SPDR=0xff;
}
//-------------------------------------------------------
void Enter_Idle_Mode(void)
{
        PCON |= 0x01;
}
//-------------------------------------------------------
void SPI_Initial(void)
{
clr_SPR0; //SPI clock = Fosc/16
clr_SPR1;
clr_MSTR; //SPI in Slave mode
clr_LSBFE; //MSB first
clr_CPOL;
```

```
set_CPHA;
set_ESPI; //Enable SPI interrupt
set_SPE; //Enable SPI function
}
//-------------------------------------------------------
void main(void)
{
lcd1602_init();
lcd1602_gotoxy(0,0);
lcd1602_puts("Test SPI");
SPI_Initial();
SPDR = ERROR; //Cho thanh ghi du lieu SPI= 0xFF

EA = 1; //Interrupt global
clr_SPIF; //xoa co ngat SPI
while(1)
        {
        Enter_Idle_Mode();
        lcd1602_gotoxy(0,1);
        lcd1602_putchar(SPDR);
        }
}
```

10. PROJECT 10: Data flash

N78E055A có hỗ trợ bộ nhớ data flash với dung lượng 4K, dùng để lưu trữ dữ liệu dù không có nguồn nuôi chip. Ta chỉ có thể can thiệp mềm vào bộ nhớ data flash bằng chế độ ISP hoặc bằng phần cứng thông qua trình nạp song song.

Ở đây, ta sẽ thử demo một chương trình can thiệp vào data flash của vi điều khiển N78E055A. Chương trình sẽ kiểm tra truy cập vào bộ nhớ data flash và thông báo kết quả thông qua hiển thị led đơn trên bo mạch.

```
#include <stdio.h>
#include "N78E055A_059A_517A.h”
#include "Typedef.h"
#include "Common.h"

#define   ERASE_ALL_AP       0x22
#define   FLASH_READ_AP      0x00
#define   BYTE_PROGRAM_AP    0x21
//-------------------------------------------------------------------------------------------
void ISP_Error(void)
{
   P1 = 0x78;
   while(1);
}
```

```
//-----------------------------------------------------------------------------------------------
void Enable_ISP_Mode(void)
{
    bit EA_tmp;

    /* Enable ISP  */
    EA_tmp = EA;
    EA = 0;
    TA = 0xAA;
    TA = 0x55;
    CHPCON |= 0x01;
    EA = EA_tmp;
}
//-----------------------------------------------------------------------------------------------
void Disable_ISP_Mode(void)
{
    bit EA_tmp;

    /* Disable ISP */
    EA_tmp = EA;
    EA = 0;
    TA = 0xAA;
    TA = 0x55;
    CHPCON &= 0xFE;
    EA = EA_tmp;
}
//-----------------------------------------------------------------------------------------------
void Trigger_ISP(void)
{
    bit EA_tmp;

    EA_tmp = EA;
    EA = 0;
    TA = 0xAA;
    TA = 0x55;
    ISPTRG |= 0x01;
    EA = EA_tmp;
}
//-----------------------------------------------------------------------------------------------
void Erase_4k_Data_Flash(void)
{
     UINT16 u16Count;

     Enable_ISP_Mode();
```

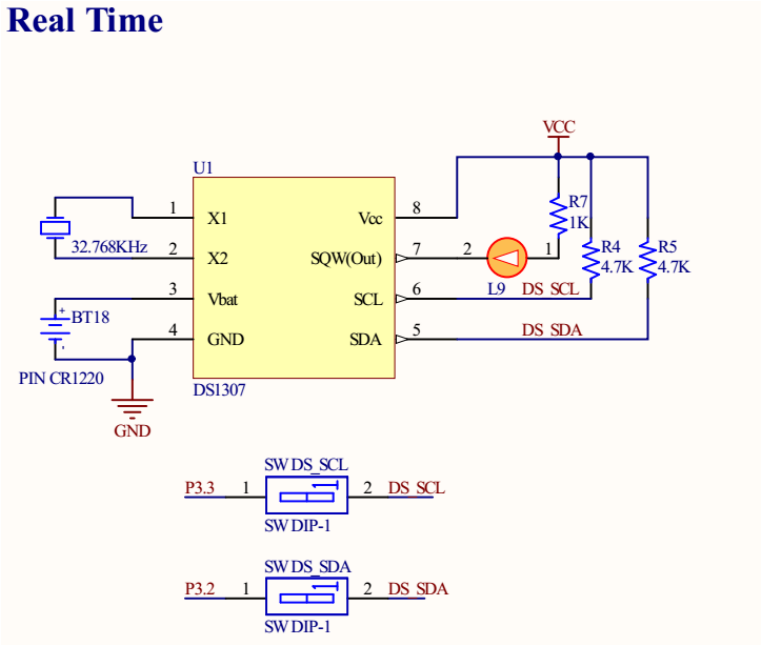
```
    P1 = 0x01;
    ISPAL = 0;
    ISPCN = ERASE_ALL_AP;

    for(u16Count=0xf0;u16Count<0x100;u16Count++)
    {
      ISPAH = u16Count;
      Trigger_ISP();
    }

    Disable_ISP_Mode();
}
//-----------------------------------------------------------------------------------------------
void Erase_4k_Data_Flash_Verify(void)
{
    UINT32 u32Count;

    Enable_ISP_Mode();
    P1 = 0x02;
    ISPAL = 0;
    ISPAH = 0xf0;
    ISPCN = FLASH_READ_AP;

    for(u32Count=0xf000;u32Count<0x10000;u32Count++)
    {
      ISPFD = 0;
      Trigger_ISP();

      if(ISPFD != 0xFF)
        ISP_Error();

      ISPAL++;
      if(ISPAL == 0)
        ISPAH++;
    }
            Disable_ISP_Mode();
}
//-----------------------------------------------------------------------------------------------
void Program_4K_Data_Flash(void)
{
    UINT32 u32Count;

    Enable_ISP_Mode();
    P1 = 0x03;
```

```
    ISPAL = 0;
    ISPAH = 0xf0;
    ISPFD = 0xFF;
    ISPCN = BYTE_PROGRAM_AP;

    for(u32Count=0xf000;u32Count<0x10000;u32Count++)
    {
        ISPFD++;
        Trigger_ISP();

        ISPAL++;
        if(ISPAL == 0)
            ISPAH++;
    }

    Disable_ISP_Mode();
}
//-----------------------------------------------------------------------------------------------
void Program_4K_Data_Flash_Verify(void)
{
    UINT32 u32Count;
    UINT8  u8Read_Data;

    Enable_ISP_Mode();
    P1 = 0x04;
    ISPAL = 0;
    ISPAH = 0xf0;;
    u8Read_Data = 0x00;
    ISPCN = FLASH_READ_AP;

    for(u32Count=0xf000;u32Count<0x10000;u32Count++)
    {

        Trigger_ISP();
        if(ISPFD != u8Read_Data)
                ISP_Error();

        ISPAL++;
        if(ISPAL == 0)
            ISPAH++;
        u8Read_Data++;
    }

    Disable_ISP_Mode();
```

```
    }
    //-----------------------------------------------------------------------------------------
    void main (void)
    {
        InitialUART0_Timer1(9600);          /* 9600 Baud Rate @ 11.0592MHz */


        Erase_4k_Data_Flash();
        Erase_4k_Data_Flash_Verify();
        Program_4K_Data_Flash();
        Program_4K_Data_Flash_Verify();

        P1 = 0x66;

        while(1);
}
```

11. PROJECT 11: Giao tiếp DS1307

DS1307 là 1 ic thông dụng và tương đối khỏe trong việc ứng dụng đồng hồ thời gian thực (RTC-real time clock), sử dụng thông qua giao tiếp I2C đơn giản với 1 dây SDA và 1 dây SCL.



Kit At89S52 v3 có tích hợp đồng hồ thời gian thực sử dụng DS1307 thông qua giao thức kết nối I2C. Để có thể sử dụng phần cứng này, ta cần thêm các file Soft_I2c.h, DS1307.h vào thư mục Include, các file Soft_I2c.c và DS1307.c vào thư mục common. Sử dụng các bước thêm file thư viện như đã hướng dẫn để có thể giao tiếp với IC DS1307. Các file Soft_I2c chứa các lệnh liên quan đến giao tiếp I2C và các file DS1307 chứa các hàm giao tiếp ghi và đọc dữ liệu thời gian cho ic DS1307.

Sau đây là nội dung các file thêm vào.

- Soft_I2c.h:

```
#ifndef   _SOFT_I2C_H_
#define _SOFT_I2C_H_
// Init
void Soft_I2c_Init();
// Master generate Start signal
void Soft_I2c_Start();
// Master generate Stop signal
void Soft_I2c_Stop();
// Write data to Slaver, and get ACK/NACK from Slaver
bit Soft_I2c_Write(unsigned char dat);
// Read data from Slaver
unsigned char Soft_I2c_Read(bit ack);
#endif
```

- DS1307.h:

```
#ifndef_RTC_DS1307_
#define _RTC_DS1307_

// Allow Ds1307 Run
void Ds1307_Init();

// Write a byte into address
void Ds1307_Write(unsigned char add, unsigned char dat);

// Read a byte at address
unsigned char Ds1307_Read(unsigned char add);

// Read a Hour, Minute, Second in Decimal format
// Read mode 12/24
// return AM/PM (0 - AM, 1 - PM)
bit Ds1307_Read_Time(unsigned char * hour, unsigned char * minute,
unsigned char * second, unsigned char * mode);

// Write Hour, Minute, Second in Decimal format
// Write Mode 12/24
// Write AM/PM (0 - AM, 1 - PM)
void Ds1307_Write_Time(unsigned char hour, unsigned minute,
unsigned char second, unsigned char mode, bit apm);

// Read day, date, month, year from Ds1307
void Ds1307_Read_Date(unsigned char * day, unsigned char * date,
unsigned char * month, unsigned char * year);
```

```
// Write day, date, month, year into Ds1307
void Ds1307_Write_Date(unsigned char day, unsigned char date,
unsigned char month, unsigned char year);

// Write array of byte begin at address
void Ds1307_Write_Bytes(unsigned char add, unsigned char * buff,
unsigned char len);

// Read array of byte into buff at add
void Ds1307_Read_Bytes(unsigned char add,unsigned char * buff,
unsigned char len);

#endif
```

- Soft_I2c.c:

```
#include "N78E055A_059A_517A.h"
#include"Soft_I2c.h"
#include"intrins.h"

#ifdef  USE_SOFT_I2C_DELAY
#message "Soft I2C - USE_SOFT_I2C_DELAY"
#define Soft_I2c_Delay()    {_nop_();_nop_();_nop_();_nop_();_nop_();}
#else
#define Soft_I2c_Delay()
#endif

#define SOFT_I2C_SCL P33
#define SOFT_I2C_SDA P32

/*-------------------------------------*-
Prototype for Local Function
-*-------------------------------------*/
bit Soft_I2c_Get_Ack();
void Soft_I2c_Ack();
void Soft_I2c_Nak();

/*-------------------------------------*-
Soft_I2c_Init
-*-------------------------------------*/
void Soft_I2c_Init()
{
        SOFT_I2C_SCL=1;
SOFT_I2C_SDA=1;
```

```
}

/*-------------------------------------*-
Soft_I2c_Start
-*-------------------------------------*/
void Soft_I2c_Start()
{
SOFT_I2C_SCL = 1;
Soft_I2c_Delay();
SOFT_I2C_SDA = 0;
Soft_I2c_Delay();
SOFT_I2C_SCL = 0;
}

/*-------------------------------------*-
Soft_I2c_Get_Ack - Local Function
- return
        0 - ACK
        1 - NAK
-*-------------------------------------*/
bit Soft_I2c_Get_Ack()
{
bit result;
   SOFT_I2C_SDA = 1;
Soft_I2c_Delay();
   SOFT_I2C_SCL = 1;
Soft_I2c_Delay();
result = SOFT_I2C_SDA;
   SOFT_I2C_SCL = 0;
return result;
}

/*-------------------------------------*-
Soft_I2c_Write
- return
        0: ACK - No Error
        1: NAK - Error
-*-------------------------------------*/
bit Soft_I2c_Write(unsigned char dat)
{
unsigned char i;
for(i=0;i<8;i++)
{
   SOFT_I2C_SDA = (bit)(dat&0x80);
```

```
    SOFT_I2C_SCL = 1;
        Soft_I2c_Delay();
        SOFT_I2C_SCL = 0;
    dat<<=1;
}
return(Soft_I2c_Get_Ack());
}

/*-------------------------------------*-
Soft_I2c_Ack - Local Function
-*-------------------------------------*/
void Soft_I2c_Ack()
{
SOFT_I2C_SDA = 0;
Soft_I2c_Delay();
SOFT_I2C_SCL = 1;
Soft_I2c_Delay();
    SOFT_I2C_SCL = 0;
}

/*-------------------------------------*-
I2C_Nak - Local Function
-*-------------------------------------*/
void Soft_I2c_Nak()
{
    SOFT_I2C_SDA = 1;
Soft_I2c_Delay();
    SOFT_I2C_SCL = 1;
Soft_I2c_Delay();
    SOFT_I2C_SCL = 0;
}

/*-------------------------------------*-
I2C_Read function
- Tham so
        ack: 0 - Master tao NAK sau khi truyen
                1 - Master tao ACK sau khi truyen
-*-------------------------------------*/
unsigned char Soft_I2c_Read(bit ack)
{
    unsigned char i, dat=0;
    for(i=0;i<8;i++)
{
    SOFT_I2C_SDA = 1;
```

```
            Soft_I2c_Delay();
        SOFT_I2C_SCL = 1;
            Soft_I2c_Delay();
        dat <<= 1;
            if(SOFT_I2C_SDA)
            {
                dat |= 0x01;
            }
            SOFT_I2C_SCL = 0;
        }
    if(ack)
    {
            Soft_I2c_Ack();
    }
    else
    {
            Soft_I2c_Nak();
    }
        return dat;
    }

    /*-------------------------------------*-
    Soft_I2c_Stop function
    -*-------------------------------------*/
    void Soft_I2c_Stop()
    {
    SOFT_I2C_SDA = 0;
    Soft_I2c_Delay();
    SOFT_I2C_SCL = 1;
    Soft_I2c_Delay();
    SOFT_I2C_SDA = 1;
    }
```

- DS1307.c:

```
#include "N78E055A_059A_517A.h"
#include"Soft_I2C.h"
#include"DS1307.h"
void Ds1307_Init()
{
 unsigned char tmp;
tmp = Ds1307_Read(0x00);
tmp &= 0x7F;
Ds1307_Write(0x00,tmp);
```

```
}
void Ds1307_Write(unsigned char add, unsigned char dat)
{
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(add);
Soft_I2c_Write(dat);
Soft_I2c_Stop();
}
unsigned char Ds1307_Read(unsigned char add)
{
unsigned char dat;
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(add);
Soft_I2c_Start();
Soft_I2c_Write(0xD1);
dat = Soft_I2c_Read(0);
Soft_I2c_Stop();
return dat;
}
/*-------------------------------------*-
Ds1307_Read_Time
- Description
        Get hour, minute, second in BCD format
- Paramaters
        &hour       -
        &minute -
        &second -
        &mode       - Mode 12/24h (12 or 24)
- return mode 24/12
        0 - AM
        1 - PM
-*-------------------------------------*/
bit Ds1307_Read_Time(unsigned char * hour, unsigned char * minute,
unsigned char * second, unsigned char * mode)
{
unsigned char h_tmp, m_tmp, s_tmp;
bit am_pm;
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(0x00);
Soft_I2c_Start();
Soft_I2c_Write(0xD1);
```

```
s_tmp = Soft_I2c_Read(1);
m_tmp = Soft_I2c_Read(1);
h_tmp = Soft_I2c_Read(0);
Soft_I2c_Stop();
s_tmp &= 0x7F;
*second = (s_tmp>>4)*10+(s_tmp&0x0F);
m_tmp &= 0x7F;
*minute = (m_tmp>>4)*10+(m_tmp&0x0F);

if(h_tmp & 0x40)            // Mode 12h
{
        *mode = 12;
        if(h_tmp & 0x20)
        {
                am_pm = 1;          // PM
        }
        else
        {
                am_pm = 0;
        }
        h_tmp &= 0x1F;
        *hour = (h_tmp>>4)*10+(h_tmp&0x0F);
}
else
{
        *mode = 24;
        h_tmp &= 0x3F;
        *hour = (h_tmp>>4)*10+(h_tmp&0x0F);
        if(*hour<12)
        {
                am_pm = 0;          // AM
        }
        else
        {
                am_pm = 1;
        }
}
return am_pm;
}

/*-----------------------------------------------------------------*-
        Ds1307_Write_Time
----------------------------------------
        - Description
```

```
                Write Hour, minute, second, mode, am/pm into Ds1307
        - Parameters
                hour, minute, second in decimal format
                mode: 12/14
                apm: 0 - AM, 1 - PM
-*-----------------------------------------------------------------*/
void Ds1307_Write_Time(unsigned char hour, unsigned minute,
unsigned char second, unsigned char mode, bit apm)
{
second = ((second/10)<<4)|(second%10);
minute = ((minute/10)<<4)|(minute%10);
hour   = ((hour  /10)<<4)|(hour  %10);
if(mode==12)
{
        hour |= 0x40;
        if(apm)                 // PM
        {
                hour |= 0x20;
        }
}
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(0x00);
Soft_I2c_Write(second);
Soft_I2c_Write(minute);
Soft_I2c_Write(hour);
Soft_I2c_Stop();
}

/*-------------------------------------*-
Ds1307_Read_Time
- Description
        Read day, date, month, year from Ds1307
- Parameters
        &date, &day, &month, &year
-
-*-------------------------------------*/
void Ds1307_Read_Date(unsigned char * day, unsigned char * date,
unsigned char * month, unsigned char * year)
{
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(0x03);
Soft_I2c_Start();
```

```
Soft_I2c_Write(0xD1);
*day  = Soft_I2c_Read(1);
*date = Soft_I2c_Read(1);
*month= Soft_I2c_Read(1);
*year = Soft_I2c_Read(0);
Soft_I2c_Stop();
*day &= 0x07;
*date &= 0x3F;
*date = (*date>>4)*10 + (*date & 0x0F);
*month &= 0x1F;
*month = (*month>>4)*10 + (*month & 0x0F);
*year = (*year>>4)*10 + (*year & 0x0F);
}
/*-------------------------------------*-
Ds1307_Write_Date
- Description
        Write day, date, month, year into Ds1307
- Parameters
        day, date, month, year
-*-------------------------------------*/
void Ds1307_Write_Date(unsigned char day, unsigned char date,
unsigned char month, unsigned char year)
{
date    = ((date/10)<<4)  | (date%10);
month = ((month/10)<<4) | (month%10);
year    = ((year/10)<<4)  | (year%10);
Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(0x03);
Soft_I2c_Write(day);
Soft_I2c_Write(date);
Soft_I2c_Write(month);
Soft_I2c_Write(year);
Soft_I2c_Stop();
}
/*-------------------------------------*-
Ds1307_Write_Bytes
- Description
        Write array of byte begin at address
- Parameters
        add: Start Address
        buff: Pointer to Write Buffer
        len:  Number of byte to Write
-*-------------------------------------*/
```

```
void Ds1307_Write_Bytes(unsigned char add, unsigned char * buff,
unsigned char len)
{
unsigned char i=0;

Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(add);
for(i=0;i<len;i++)
{
        Soft_I2c_Write(buff[i]);
}
Soft_I2c_Stop();
}

/*-------------------------------------*-
Ds1307_Read_Bytes
- Description
        Read array of byte into buff at add
- Parameters
        add: Start Address
        buff: Pointer to Read Buffer
        len:  Number of byte to Read
-*-------------------------------------*/
void Ds1307_Read_Bytes(unsigned char add,unsigned char * buff,
unsigned char len)
{
        unsigned char i;

Soft_I2c_Start();
Soft_I2c_Write(0xD0);
Soft_I2c_Write(add);
Soft_I2c_Start();
Soft_I2c_Write(0xD1);
for(i=0;i<len-1;i++)
{
        buff[i]  = Soft_I2c_Read(1);
}
buff[i]  = Soft_I2c_Read(0);
Soft_I2c_Stop();
}
```

VI. LƯU Ý TÀI LIỆU

Trong quá trình biên soạn tài liệu, khó tránh khỏi các sai sót. Mong sự phản hồi tích cực từ người đọc để xây dựng một nguồn tài liệu mở đáng tin cậy.

Mọi thông tin, xin liên hệ về:

Công ty TNHH Giải pháp TULA - TULA Solution Co., Ltd

TRỤ SỞ CHÍNH (HÀ NỘI)

VP giao dịch: Số 6 Ngõ 23 Đình Thôn, Mỹ Đình, Nam Từ Liêm, Hà Nội

Tel.: 04. 39655633, Hotline: 0912612693, Fax: 04. 39655633

E-mail: info@tula.com / info (at) tulaso.com

ĐẠI DIỆN KHU VỰC MIỀN NAM

Địa chỉ: Số 273 Đường Điện Biên Phủ, Phường 7, Quận 3, TP. HCM

Tel.: 0912 612 693, Fax: 04. 39655633

E-mail: hcm@ tula.vn