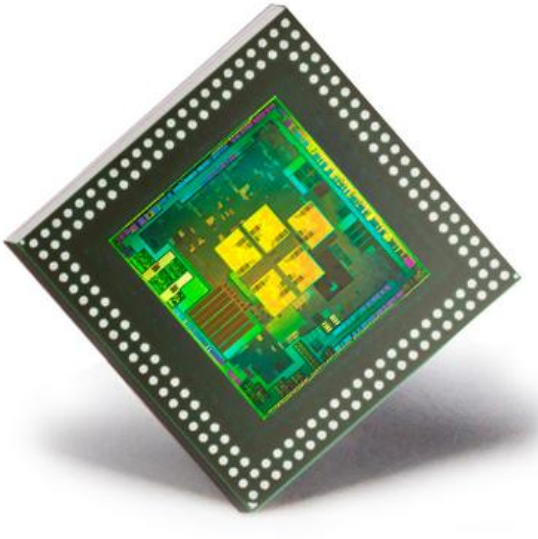
LẬP TRÌNH CĂN BẢN ARM STM32F103C8T6

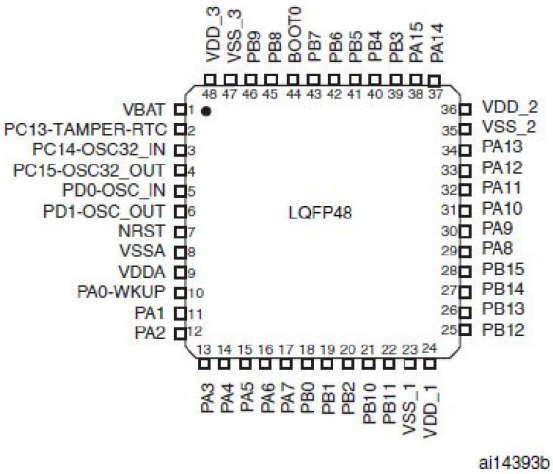


By

Họ tên: Nguyễn Ngọc Hà

TpHCM, Tháng 5, 2014

VI ĐIỀU KHIỂN ARM STM32F103C8T6

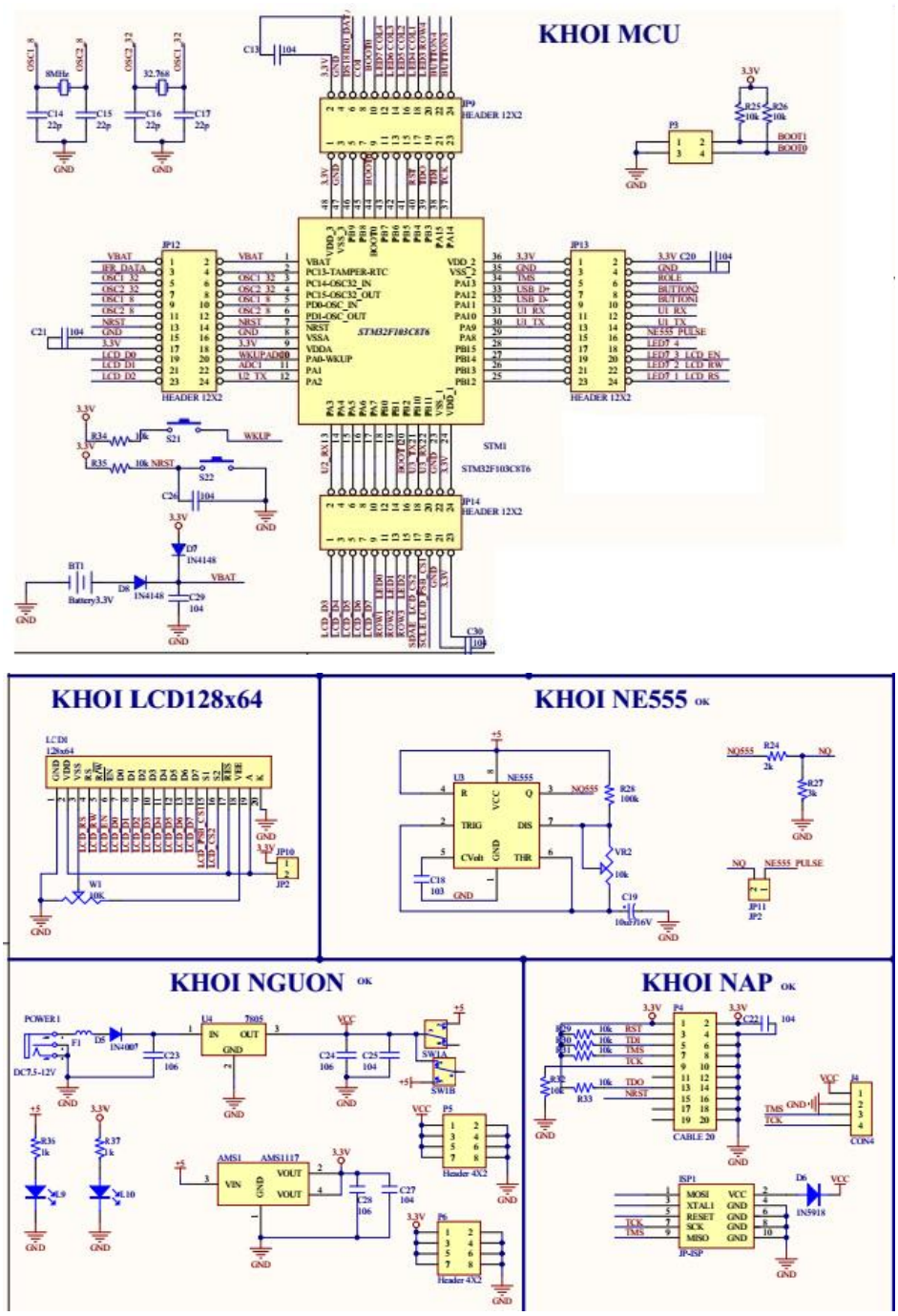


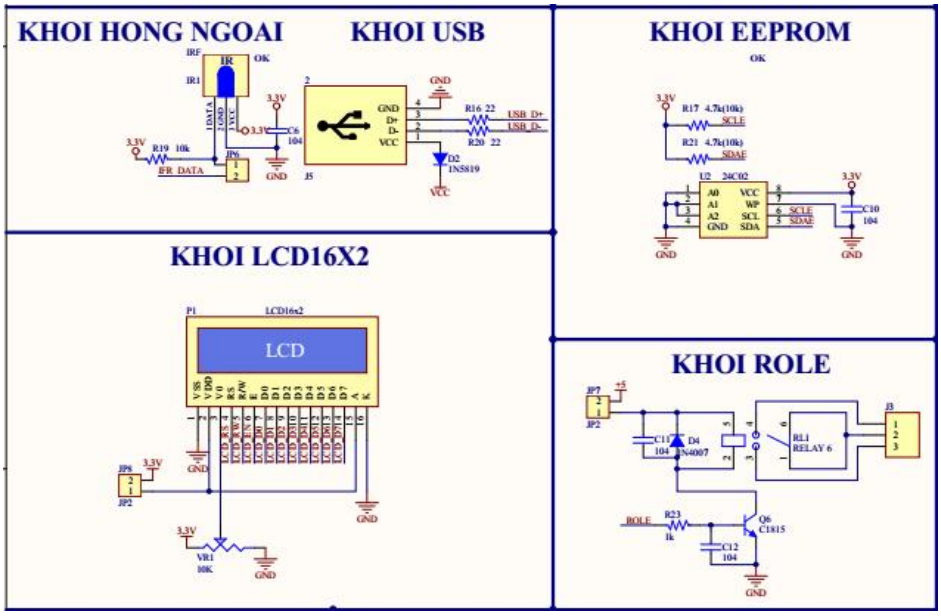
Manufacturer Part Number	STM32F103C8T6
Description	MCU ARM 64KB FLASH MEM 48-LQFP
Vendor	STMicroelectronics
Category	Integrated Circuits (ICs)
Program Memory Size	64KB (64K x 8)
RAM Size	20K x 8
Number of I /O	37
Package / Case	48-LQFP
Speed	72MHz
Oscillator Type	Internal
Packaging	Tray
Program Memory Type	FLASH
EEPROM Size	-

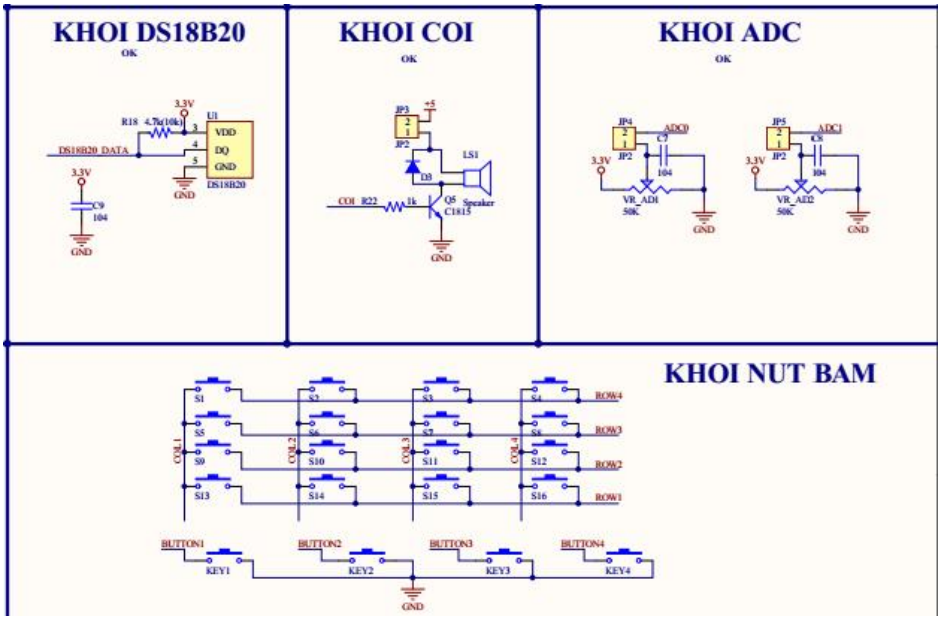
Core Processor	ARM® Cortex -M3
Data Converters	A/D 10x12b
Core Size	32-Bit
Operating Temperature	-40°C ~ 85°C
Connectivity	CAN, I²C, IrDA, LIN, SPI, UART/USART, USB
Peripherals	DMA, Motor Control PWM, PDR, POR, PVD, PWM, Temp Sensor, WDT
Voltage - Supply (Vcc/Vdd)	2 V ~ 3.6 V
Lead Free Status	Lead Free
RoHS Status	RoHS Compliant
Other Names	STM32F103C8T6 STM32F103C8T6 497 6063 ND 4976063ND 497-6063

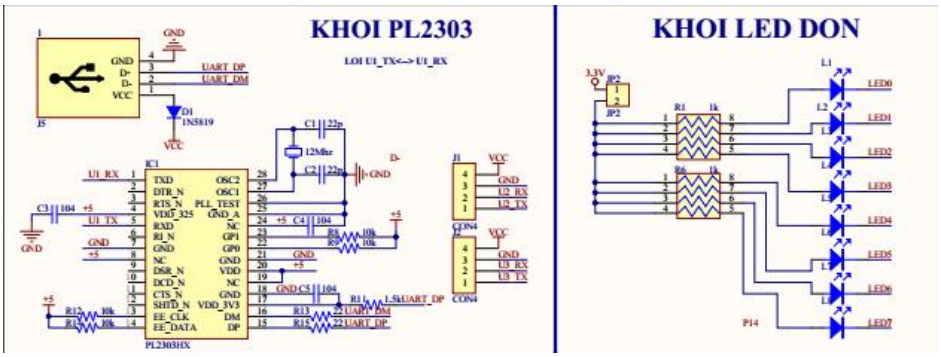
TỔNG QUAN

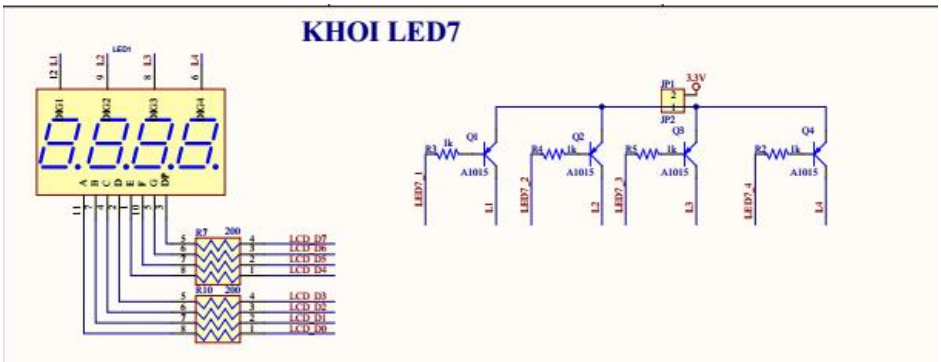
SƠ ĐỒ NGUYÊN LÝ KIT STM32F103




KHOI HONG NGOAI
KHOI USB
KHOI EEPROM
KHOI LCD16X2
KHOI ROLE


KHOI DS18B20
KHOI COI
KHOI ADC
KHOI NUT BAM


KHOI PL2303
KHOI LED DON



CÔNG CỤ HỖ TRỢ

Để bắt đầu với lập trình vi xử lý ARM STM32 trên kít STM32F103 chúng ta cần một số công cụ sau:

Driver PL2303: dùng kết nối với Vi xử lý thông qua cổng UART và nạp code cho chíp

http://bit.ly/1mXUzxk

Flash BootLoader for ARM: Nạp code cho chíp thông qua bootloader

http://bit.ly/1sUf25T

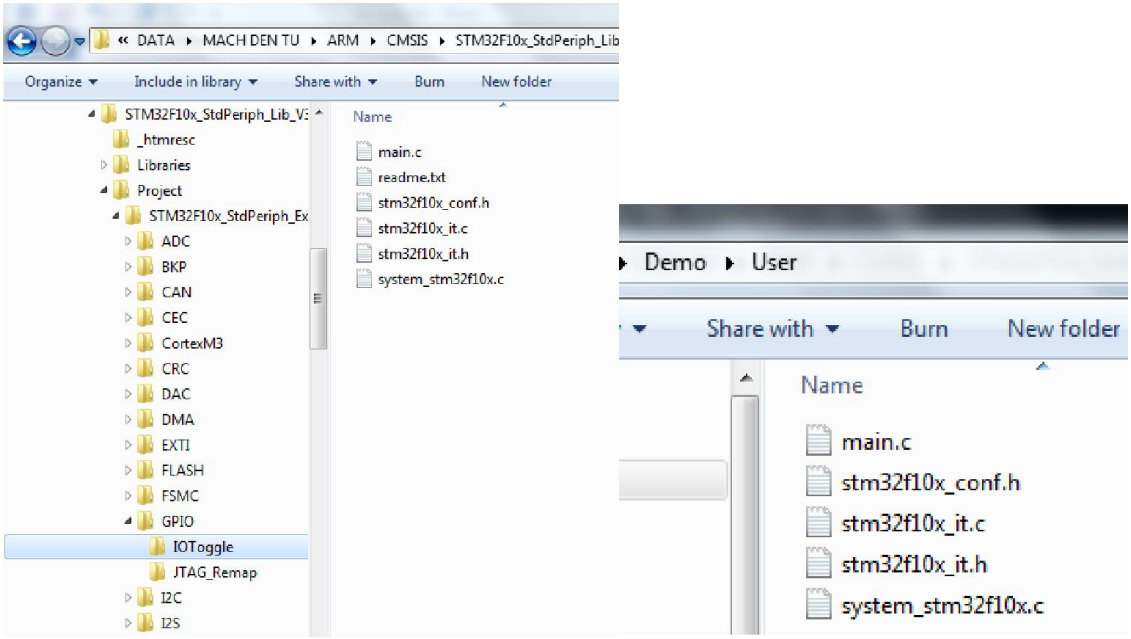
KeilC MDK : lập trình C cho dòng ARM . (bản này khác với bản Keil C chúng ta thường cài để lập trình cho VDK 8051, nếu ai chưa có thể cài thêm MDK để lập trình cho cả 8051 và ARM nhé) Link tại trang chủ : http://www.keil.com/arm/mdk.asp

Thư viện CMSIS : http://www.ulozto.net/x2JFvXv/stm32f10x-stdperiph-lib-v3-5-0-zip

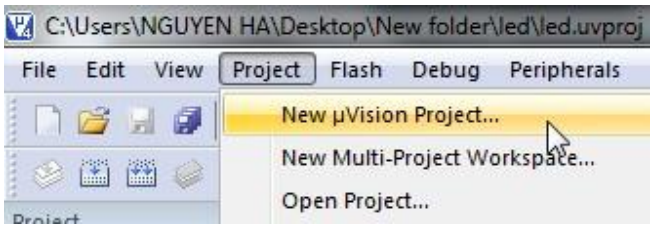
Cách cài đặt có thể tham khảo trên mạng

TẠO 1 PROJECT MỚI

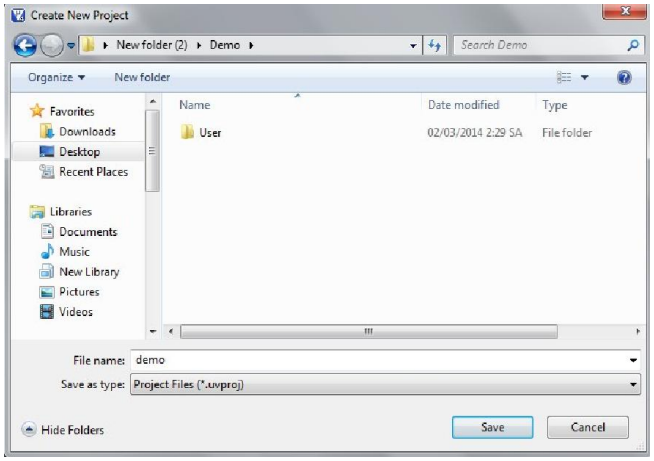
1. Download và giải nén thư viện CMSIS trên về ta có thư mục **STM32F10x_StdPeriph_Lib_V3.5.0** , trong này ta chú ý 2 thư mục chính là **Libraries** và **Project**
2. Tạo một thư mục mới để tiện quản lý và sử dụng Project. Copy thư mục **Library** ở trên cùng với thư mục mới tạo. trong thư mục mới tạo thêm một thư mục **User** để chứa những file do người dùng tạo ra. Copy các file có trong ***…\STM32F10x_StdPeriph_Lib_V3.5.0\Project\STM32F10x_StdPeriph_Examples\GPIO\IOToggle*** vào thư mục **User**



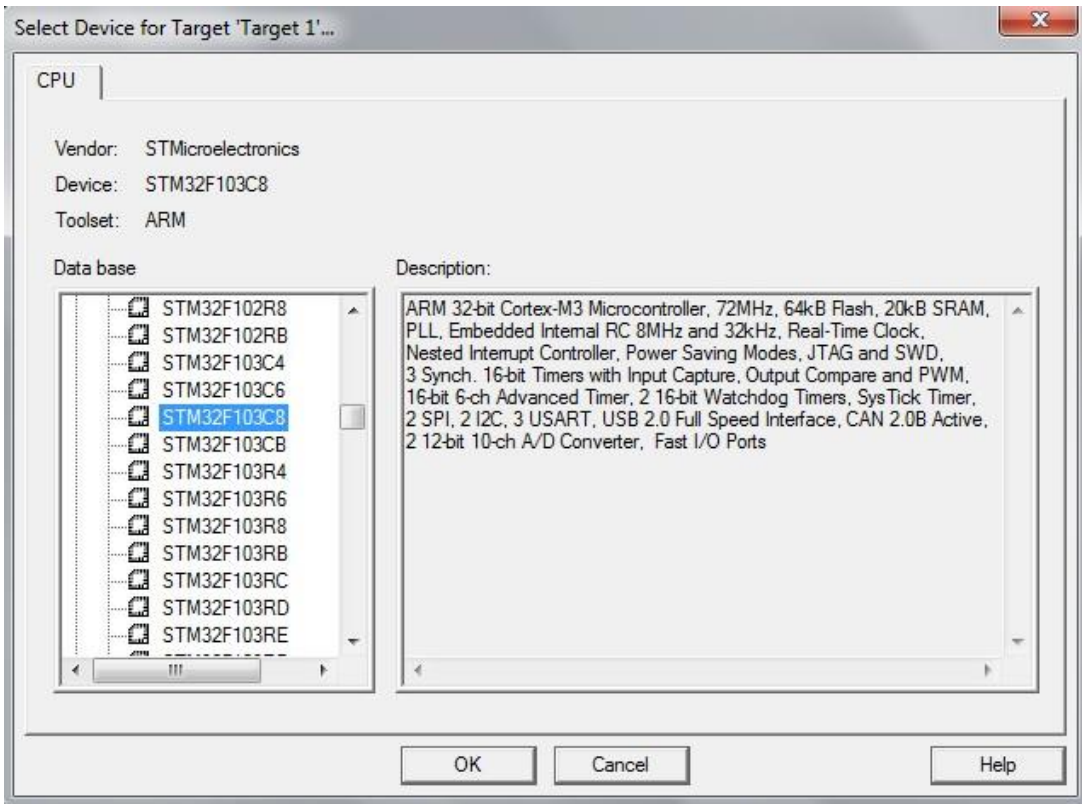
3. Mở Keil C lên và tạo một Project mới



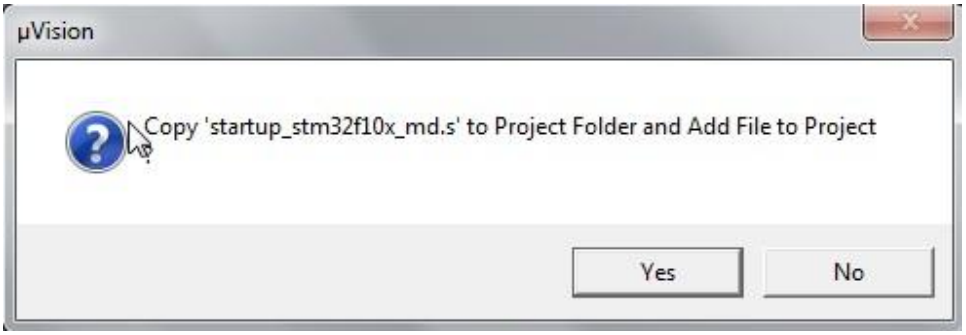
- Ở đây chúng ta tạo thư mục là **Demo**



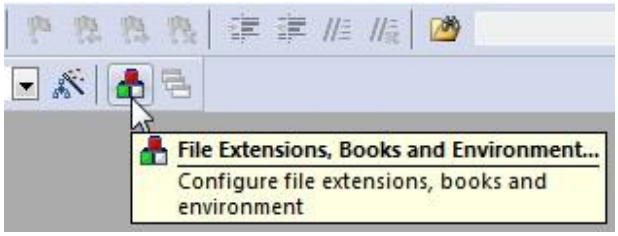
Hiện cửa sổ chọn Chip. Ở đây chọn **STMicroelectronics**. Chọn chip **STM32F103C8**

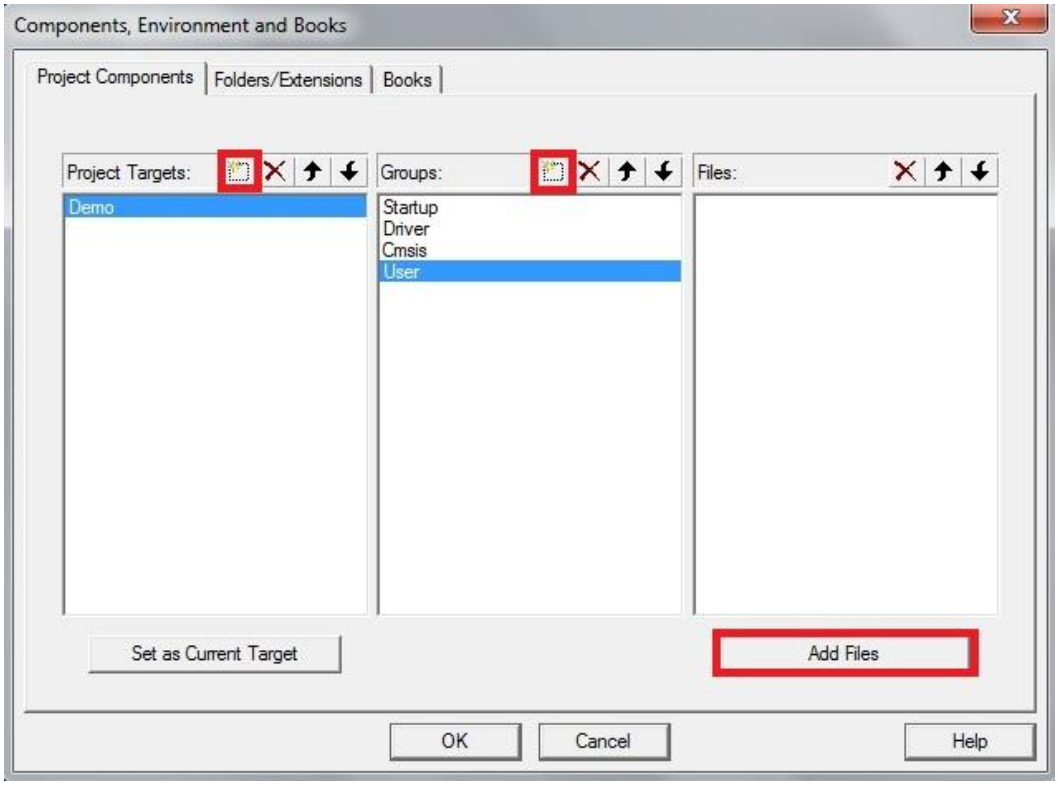


Cửa sổ mới hiện ra, chọn **No** vì không cần thiết, chúng ta sẽ add sau



Trong Project mới , nhấp chuột vào **Target**

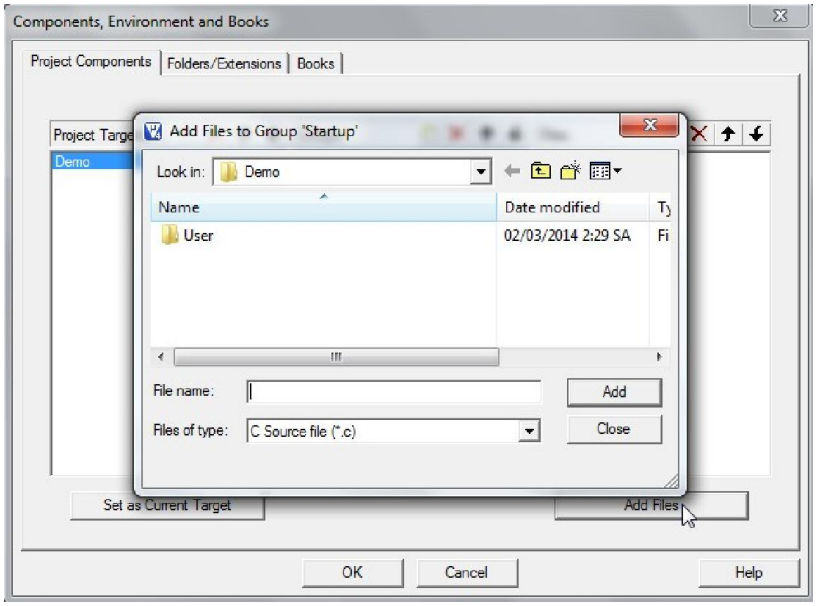




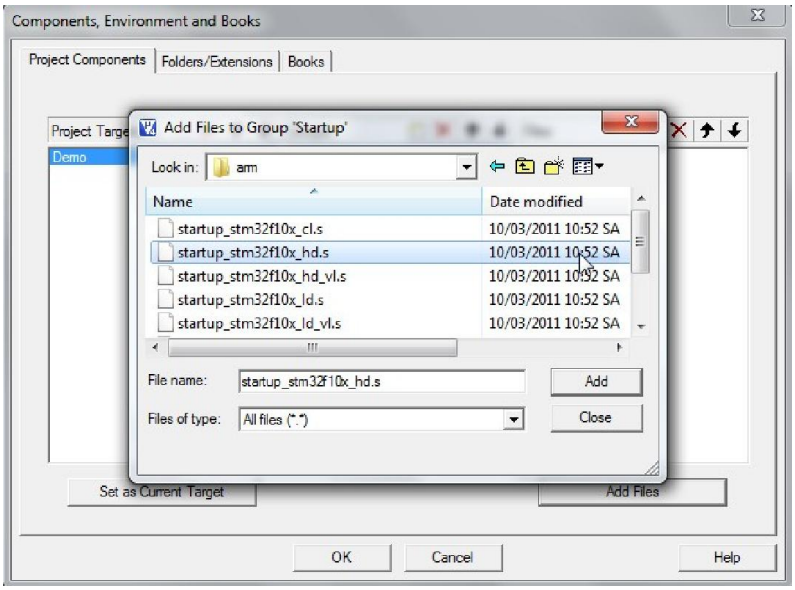
Nhấn vào ô vuông để tạo tên mới cho Project và ô vuông thứ hai để tạo các Group. Như trên là : **Startup**, **Driver**, **Cmsis**, **User**. Chọn **add files** để add một số file vào group.

Các file cần add đều nằm trong thư mục **Library**

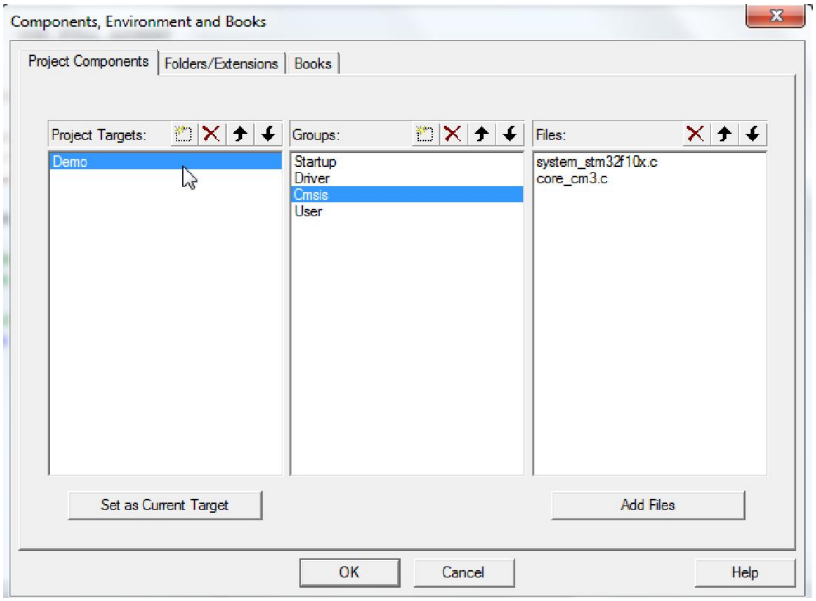
- Group **User** : add các file trong mục **User** vừa tạo ở trên



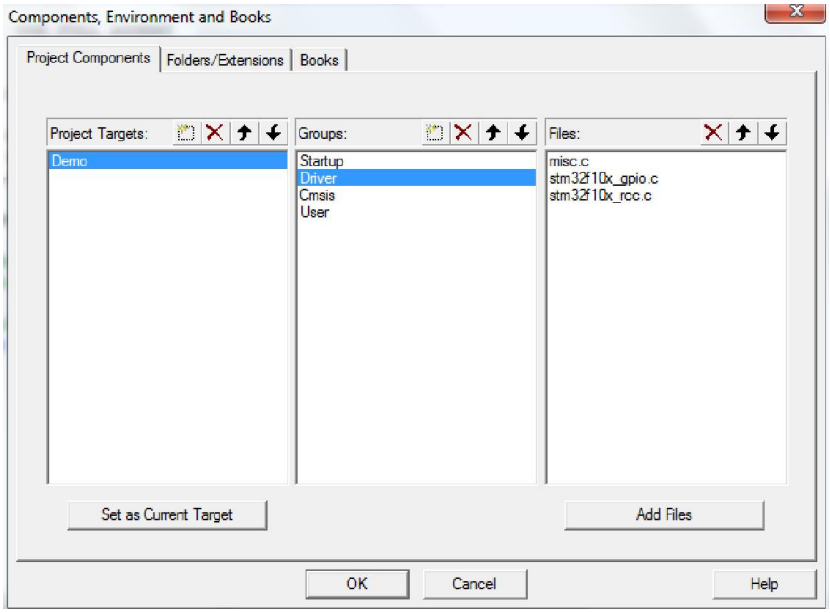
- Groups **Stratup** : add file **starup_stm32f103_hd.s** . Đường dẫn : ***STM32F10x_StdPeriph_Lib_V3.5.0\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x\startup\arm***



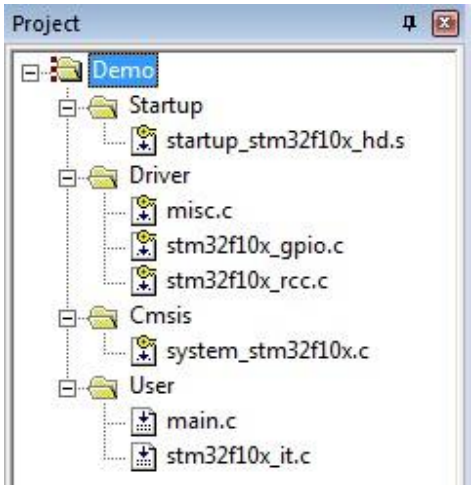
- Groups **Cmsis**: add các file **core_cm3.c** (*.\STM32F10x_StdPeriph_Lib_V3.5.0\Libraries\CMSIS\CM3\CoreSupport*), **system_stm32f10x.c** (*STM32F10x_StdPeriph_Lib_V3.5.0\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x*)



- Groups **Drive**: add các file driver cần cho Project : *STM32F10x_StdPeriph_Lib_V3.5.0\Libraries\STM32F10x_StdPeriph_Driver\src*

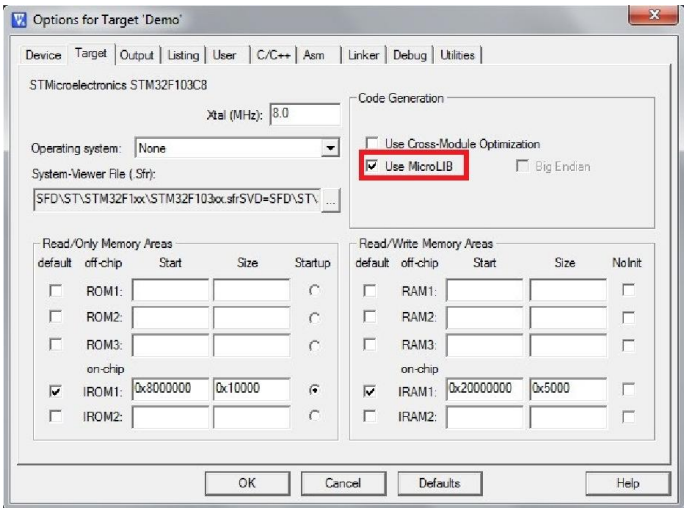


Nhấn **OK** để hoàn thành. Project của chúng ta đày đủ như hình dưới:

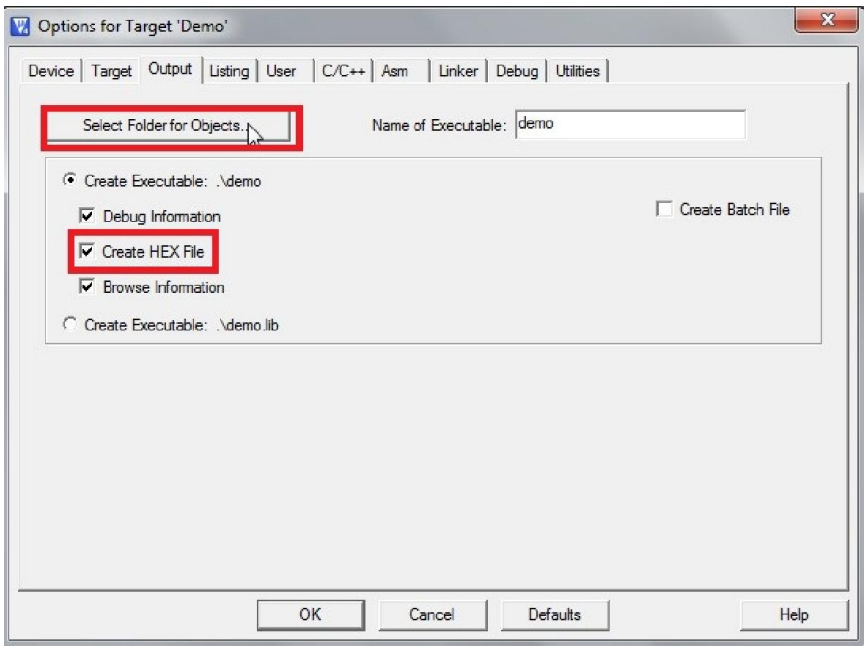


4. Tiếp theo là cấu hình cho Project
5. Chọn **Target Options** để cấu hình

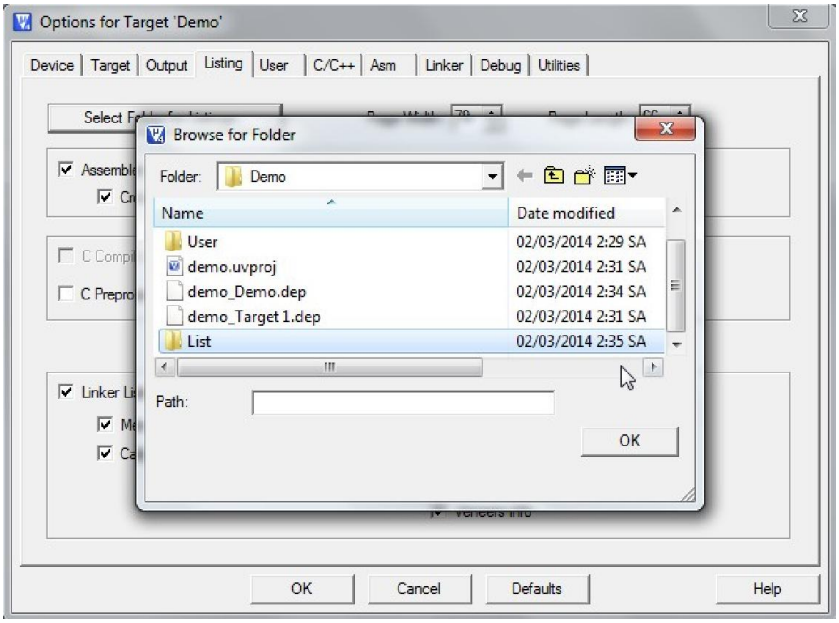
Ở tab **target** , đánh dấu chọn **Use MicroLIB**



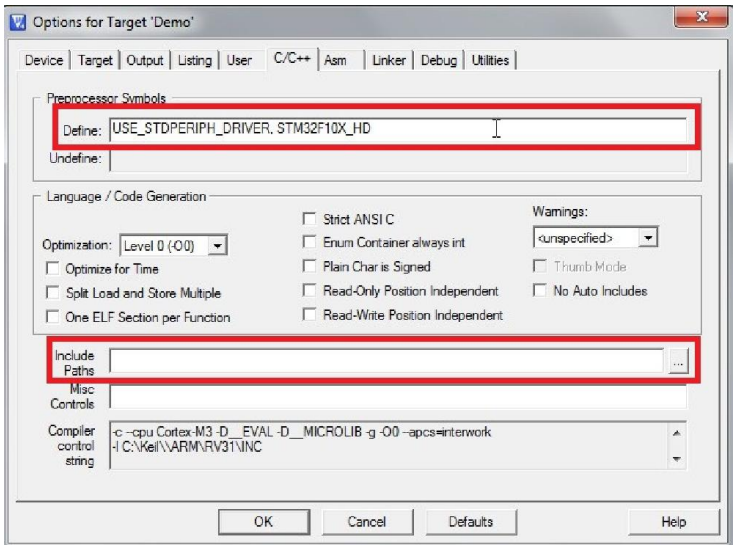
Tab **Output** : đánh dấu chọn **Create HEX File** để tạo file HEX nạp cho VDK



Chọn **Select Folder for Objects**… Và tạo một thư mục **Obj** , tương tự với tab **Listing**



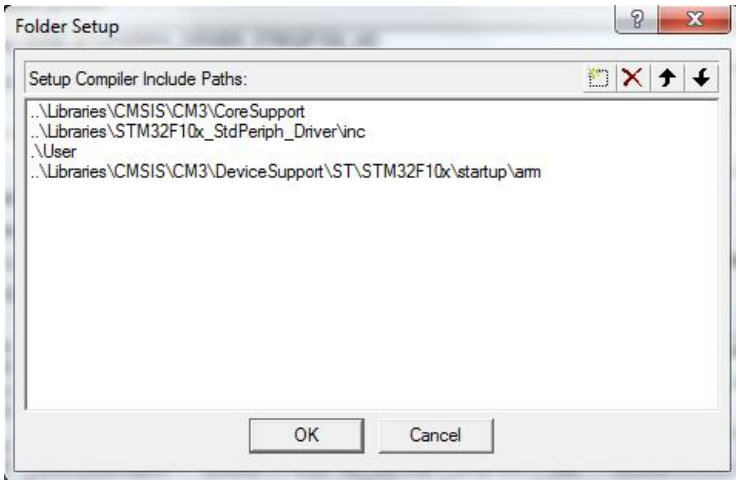
Tab **C/C++**: tại dòng **Deline** gõ vào : **USE_STDPERIPH_DRIVER, STM32F10X_HD**



USE_STDPERIPH_DRIVER : Nằm trong stm32f10x.h, khai báo sử dụng thư viện bên ngoài

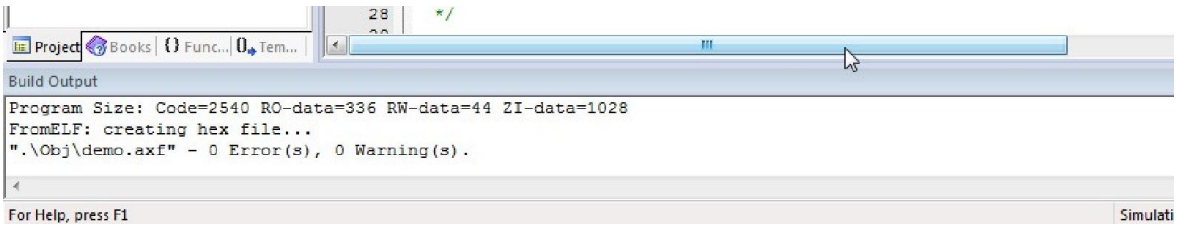
STM32F10X_HD : Flash Memory

Nhấp vào dòng **Include Paths** để cài đặt thư mục **Folder Setup** cho Project, ở bên dưới ô vuông đó là những thứ chúng ta phải add vào. Mục đích là khai báo cho trình biên dịch biết được thư viện nằm ở đâu

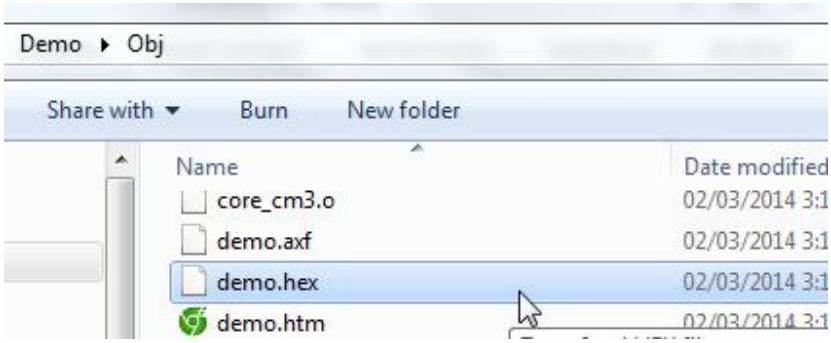


OK, Nhấn **F7** để biên dịch chương trình

Kết quả buil thành công:



File HEX ở đây:



Vậy là chúng ta đã hoàn thành xong việc tạo 1 Project mới cho **ARM STM32** dùng **KeilC**

LẬP TRÌNH GPIO BẬT TẮT LED

Trước khi bắt đầu chúng ta cần quay lại một số vấn đề trong phần 1

Chúng ta mở file stm32f10x.h lên và xem phần sau :

```
 /* #define STM32F10X_LD */    /*!< STM32F10X_LD: STM32 Low density devices */

 /* #define STM32F10X_LD_VL */ /*!< STM32F10X_LD_VL: STM32 Low density Value Line devices */

 /* #define STM32F10X_MD */    /*!< STM32F10X_MD: STM32 Medium density devices */

 /* #define STM32F10X_MD_VL */ /*!< STM32F10X_MD_VL: STM32 Medium density Value Line devices
*/

 /* #define STM32F10X_HD */    /*!< STM32F10X_HD: STM32 High density devices */

 /* #define STM32F10X_HD_VL */ /*!< STM32F10X_HD_VL: STM32 High density value line devices */

 /* #define STM32F10X_XL */    /*!< STM32F10X_XL: STM32 XL-density devices */

 /* #define STM32F10X_CL */    /*!< STM32F10X_CL: STM32 Connectivity line devices */

/*  Tip: To avoid modifying this file each time you need to switch between these

       devices, you can define the device in your toolchain compiler preprocessor.


 - Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers

   where the Flash memory density ranges between 16 and 32 Kbytes.

 - Low-density value line devices are STM32F100xx microcontrollers where the Flash

   memory density ranges between 16 and 32 Kbytes.

 - Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers

   where the Flash memory density ranges between 64 and 128 Kbytes.

 - Medium-density value line devices are STM32F100xx microcontrollers where the

   Flash memory density ranges between 64 and 128 Kbytes.

 - High-density devices are STM32F101xx and STM32F103xx microcontrollers where
```

the Flash memory density ranges between 256 and 512 Kbytes.

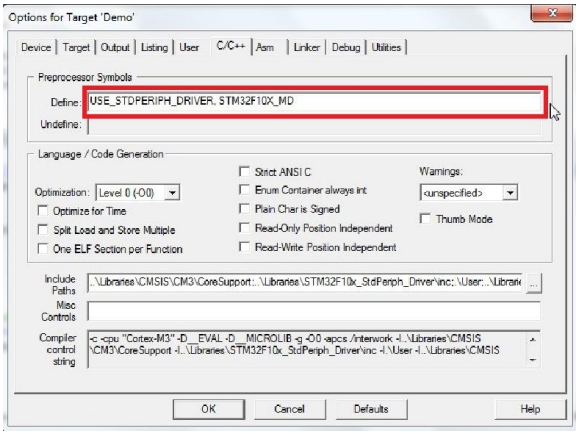
- High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

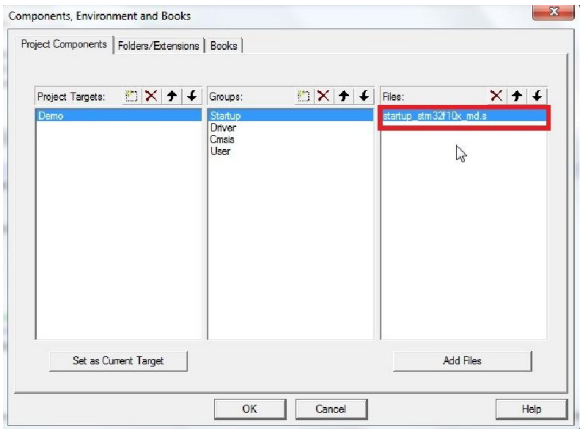
- XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 512 and 1024 Kbytes.

- Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

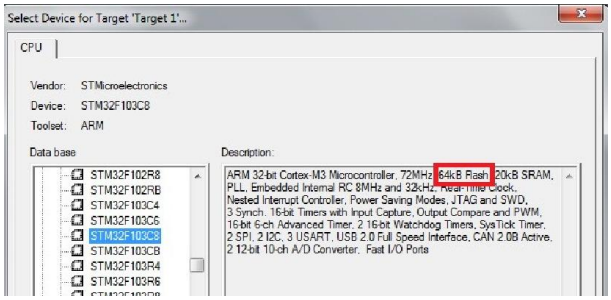
**/*

Trên là hướng dẫn chọn **define**, **file startup** cho chương trình. Tùy theo chip tương ứng mà chúng ta cần khai báo cho đúng





Muốn biết chip đang dùng thuộc loại nào thì khi khởi tạo project, lúc chọn chip có hiển thị thông tin chip, chúng ta xem **Flash** bao nhiêu để chọn **define** cho đúng



Ở phần trước mình chọn chip **STM32F103C8** , có 64kB Flash là chip **Medium-Density** nhưng lại chọn define là **STM32F10X_HD** nên để buil được và nạp code cho VDK chạy thì các bạn cần sửa lại thành **STM32F10X_MD**

OK, Vậy là đã quá rõ rồi. Bây giờ chúng ta tiếp tục với phần **GPIO**

MCU STM32F10x có nhiều loại với số lượng **IO** khác nhau. Mỗi port IO được cấu hình bởi 2 thanh ghi 32bit (**GPIOx_CRL&GPIOx_CRH**)
- **GPIOx_CRL** : cấu hình các pin từ 0→7
- **GPIO_CRH** : cấu hình các pin từ 8→15

Có 8 chế độ IO có thể lập trình cho từng pin
- Input floating
- Input pull-up
- Input pull-down
- Analog input
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Các bit mode[1:0] cấu hình chế độ input hoặc output

Mode info
00 input(mặc định khi reset)
01 output max 10MHz
10 output Max 2Mhz
11 output Max 50 MHz

Dòng F3,F4 tốc độ các chân có thể cao hơn. Thường thì mình để tốc độ cổng ở tốc độ tối đa luôn.
Các bit CNF[1:0] có ý nghĩa phụ thuộc vào trạng thái pin là input hay output
Input Mode :
CNF[1:0] info
00 analog input
01 floating input(digital)
10 input với pullup/pulldown.
11 reserver

Output Mode :
CNF[1:0] info
00 output push/pull
01 output open drain
10 alternate output push/pull
11 alternate output open drain

Chú ý : chế độ input pullup/pulldown sẽ do giá trị bit tương ứng trên thanh ghi ODR quyết định. Nếu sử dụng hàm chuẩn trong thư viện của ST thì có thể các bạn không cần biết cũng làm được.Nhưng theo mình thì nên biết xem hàm mình dùng nó tác động vào thanh ghi nào để lúc cần thì có thể gán trực tiếp cho nhanh
Các pin IO đều có dạng 5V tolerant (ngoài 2 pin chung chức năng với thạch anh đồng hồ thời gian thực) tức là có thể nối với các thiết bị dùng chuẩn 5V. Mình thường nối thêm con trở nhỏ nối tiếp với chân IO nếu nó là chế độ input (để phần điện áp dư rơi trên đó tránh gây hỏng pin IO).
Sơ đồ các pin các bạn có thể tham khảo trong datasheet.Các thanh ghi quan trọng.

- Input data register GPIOx_IDR
- Output data register GPIOx_ODR
- Bit Set/Reset register GPIOx_BSRR
- Bit Reset register GPIOx_BRR
- lock mechanism register GPIOx_LCKR

Ngoài ra còn có thanh ghi remap các chân vào ra của ngoại vi. Các bạn xem trong datasheet để hiểu rõ hơn.

Trong thư viện "*stm32f10x_gpio.h*" các pin tương ứng đã được định nghĩa sẵn để người dùng dễ sử dụng : **GPIO_Pin_x**
Các port được định nghĩa bằng tên **GPIOx** trong đó x: A,B,C,…G. Thực chất **GPIOx** có dạng con trỏ trỏ tới địa gốc của port tương ứng.
Lệnh dùng khi **Set Bit x** của **port y** : **GPIOx→BSRR = GPIO_Pin_y**
Lệnh dùng khi **Reset bit x** của **Port y** : **GPIOx→BRR =GPIO_Pin_y**
Hoặc dùng lệnh :**GPIOx→BSRR = GPIO_Pin_y <<16**
Thư viện chuẩn của ST, để bật tắt các bit, ta sử dụng hàm **GPIO_SetBit()** và **GPIO_ReSetBit().**

Bắt đầu chường trình cho GPIO :

```
#include "stm32f10x.h"
```

Khai báo thư viện stm32f10x.h

```
void delay_ms(uint32_t num);
void delay_ms(uint32_t num)
{
        uint32_t index = 0;
        for(index = (72000 * num);index !=0;index-- )
        {}
}
```

Chương trình delay với độ phân giải là 1ms, tức giá trị đặt là 72000 (72MHz). Nếu bạn muốn độ phân giải là 1us thì giá trị đặt là 72 .

```
int main(void)
{
GPIO_InitTypeDef GPIO_InitStruct;
```

Khai báo biến dữ liệu để khởi tạo modul GPIO

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
```

Cho phép xung Clock ở PortB

```
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
```

Cầu hình chon **Pin 0** và **Pin 15** ở chế độ **Push-Pull**, tốc độ **50Mhz**

```
GPIO_Init(GPIOB, &GPIO_InitStruct);
GPIO_ResetBits(GPIOB, GPIO_Pin_0|GPIO_Pin_1);
```

Khởi tạo GPIOB

```
while(1){
GPIO_ResetBits(GPIOB, GPIO_Pin_0|GPIO_Pin_1);
delay_ms(100);
```

Cho **GPIOB-Pin 0** và **Pin 1** ở mức **logic thấp**, thời gian delay là 100ms

```
GPIO_SetBits(GPIOB, GPIO_Pin_0|GPIO_Pin_1);
delay_ms(100);
```

Cho **GPIOB-Pin 0** và **Pin 1** ở mức **logic cao**, thời gian delay là 100ms

Xong, đoạn code chớp tắt led trên khá ngắn gọn và dễ hiểu. Và để kiểm tra tính chính xác thì chúng ta nạp code cho VDK xem kết quả

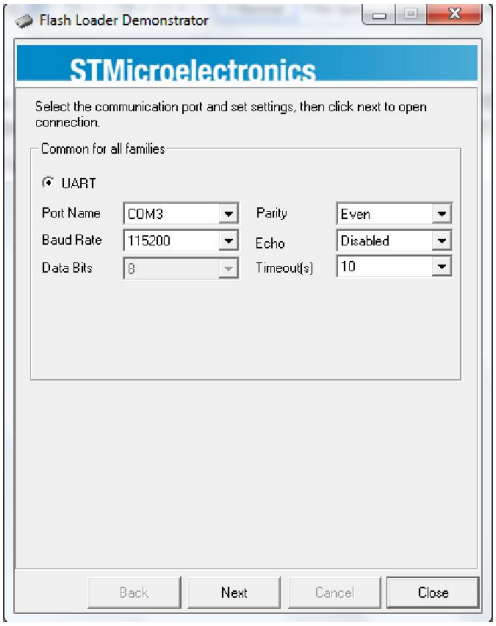
Việc nạp code cho **STM32** có nhiều cách

- **JTAG:** nạp và gỡ rối, việc dùng JTAG được thực hiện trên KeilC nên rất thuận tiện cho việc nạp code, debug , test sản phẩm,.. nhược điểm là phần cứng rườm rà.
- **SWD :** chuẩn giao tiếp 2 dây, nhỏ gọn đơn giản và chi phí thấp hơn so với JTAG
- **Bootloader :** phần cứng đơn giản, dễ thực hiện,… nhưng chỉ dùng cho việc nạp code

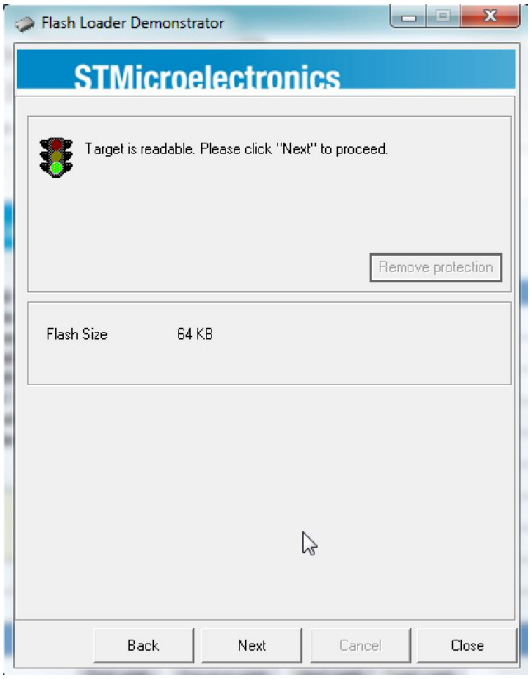
Ở đây, đa số khi các bạn mua KIT về học thì đều có hỗ trợ **JTAG** và **Bootloader** nên mình sẽ hướng dẫn các bạn nạp file HEX đã buil ở trên vào VDK thông qua **Bootloader**

Để vào được chế độ **bootloader** thì bạn phải cài đặt cho chân **BOOT0 =1** và chân **BOOT1=0**. Để chip tiếp tục chạy từ bộ nhớ Flash thì **BOOT0=0, BOOT1=1**

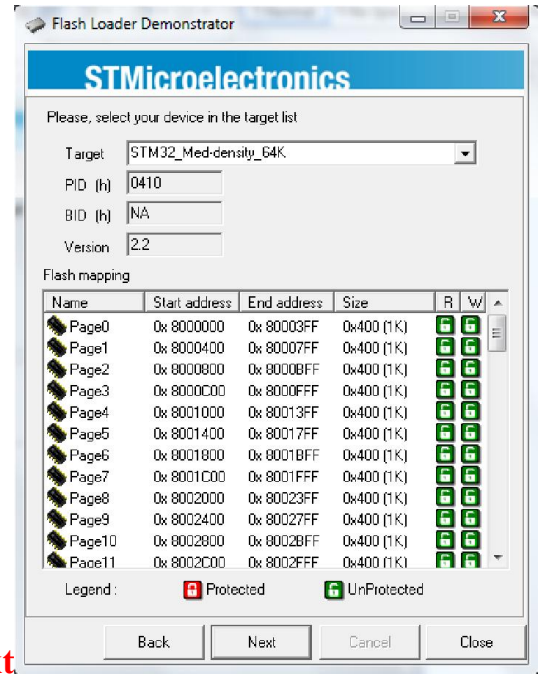
Khởi động **Flash Loader**, nếu có kết nối với KIT thì sẽ như hình:



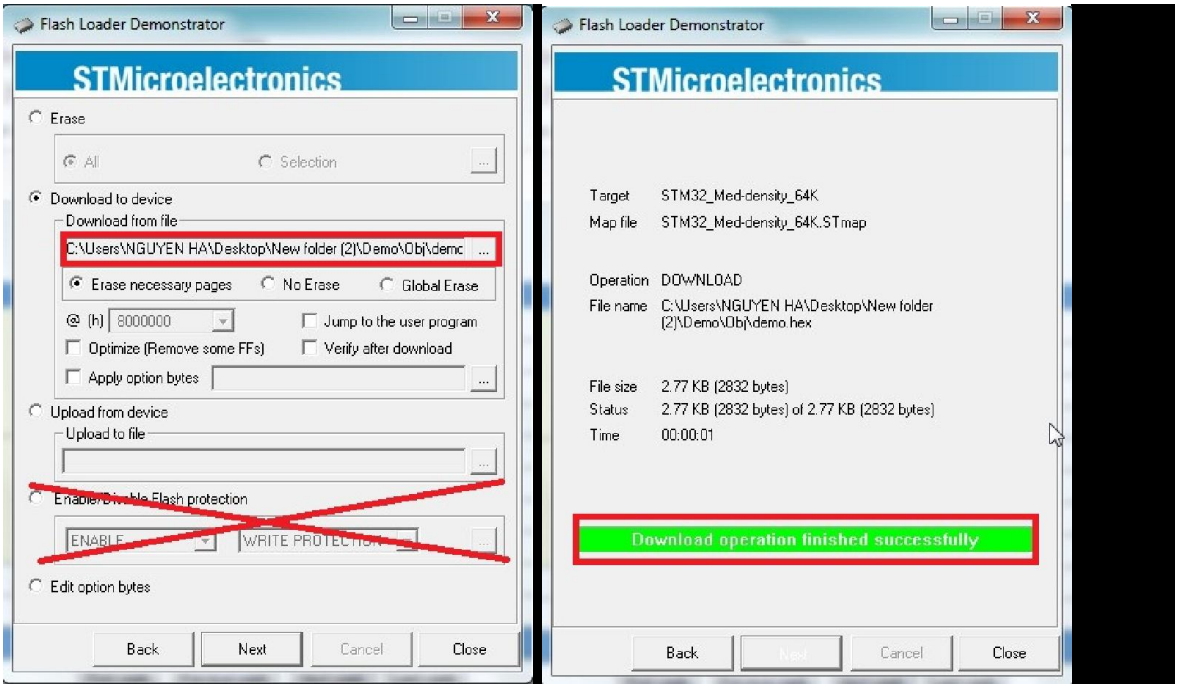
Nhấn **Next** để tiếp tục, nếu không ở bootloader thì sẽ báo lỗi, còn không thì sẽ như hình:



Next



Tiếp theo, tại **Download to device** , dẫn đến file HEX ở trên. Tuyệt đối không chọn vào **Enable/Divsble Flash protecion** nếu như không muốn khóa chip để bảo mật chương trình. Nhấn **Next** để thực hiện nạp code vào chip.



Tắt bootloader và cài đặt lại 2 chân BOOT1, BOOT0 để chạy xem nhé

Vậy là xong chương trình GPIO chớp tắt led đơn giản, coi như chúng ta đã hoàn thành được 1 project hoàn chỉnh từ việc cài đặt chương trình, khởi tạo, lập trình project và nạp code để chạy mạch thật

LẬP TRÌNH GPIO ĐỌC TRẠNG THÁI NÚT NHẤN

Tương tự như lập trình GPIO điều khiển led đơn, ta cần khai báo thêm trạng thái input cho các chân input

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11|GPIO_Pin_12|GPIO_Pin_14|GPIO_Pin_15;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Trong chương trình chính ta chỉ cần đọc trang thái của các chân tín hiệu tương ứng để điều khiển ouput ra led đơn hoặc các chức năng khác

```
GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_11); /
```

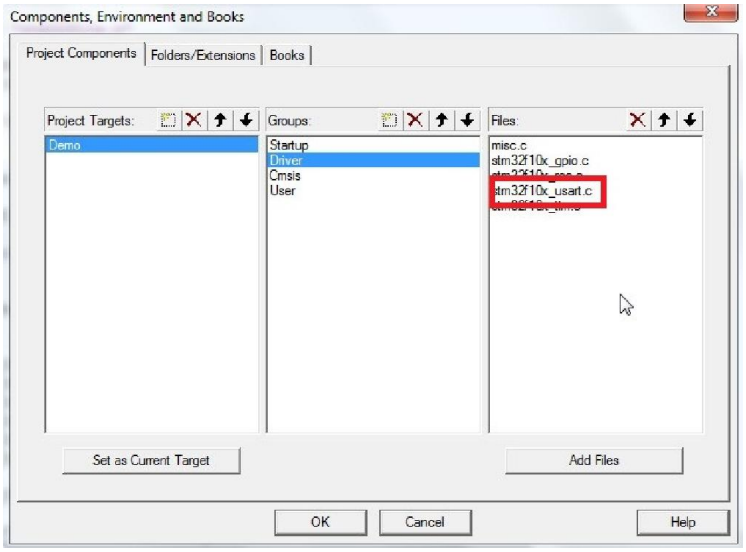
đọc trạng thái trên từng Pin

```
GPIO_ReadInputData(GPIOA);
```

đọc trạng thái in put GPIO A

1. Lập trình UART cho ARM STM32

- Tạo một Project tương tự như với lập trình GPIO , chúng ta thêm driver stm32f10x_uart.c trong thư viện CMIS vào để có thể lập trình truyền nhận dữ liệu UART



- Viết code cho chương trình chính trong hàm maic.c

Khai báo IO cho UART:

```
GPIO_Init(GPIOA, &GPIO_InitStructure);
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;                    //USART1 TX
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
 GPIO_Init(GPIOA, &GPIO_InitStructure);


 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10|GPIO_Pin_8;          //USART1 RX
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
       GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
```

```
 GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Cấu hình UART:

```
void USART1_Configuration(void)
{

 USART_InitTypeDef USART_InitStructure;
        //Kich hoat Clock USART1
 RCC_APB2PeriphClockCmd( RCC_APB2Periph_USART1, ENABLE);
        //Chon BaudRate
 USART_InitStructure.USART_BaudRate = 115200;

        //Chon do dai khung truyen
 USART_InitStructure.USART_WordLength = USART_WordLength_8b;
        //Chon stop bit
 USART_InitStructure.USART_StopBits = USART_StopBits_1;
        //Chon Parity
 USART_InitStructure.USART_Parity = USART_Parity_No;
        //Chon che do dieu khien
 USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        //Chon phuong thuc truyen nhan
 USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//
        //Cau hinh cac thong so vua lua chon
 USART_Init(USART1, &USART_InitStructure);
```

```
    //Kich hoat hoat dong cua USART
 USART_Cmd(USART1, ENABLE);
}
```

Hàm ghi dữ liệu

Hàm đọc dữ liệu từ cổng UART:

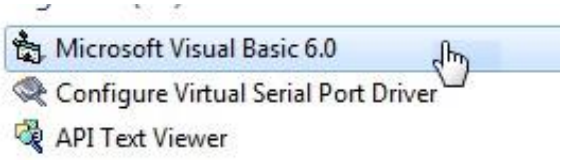
```
USART_ReceiveData(USART1);
```

Hàm gửi chuỗi dữ liệu ra cổng UART:

```
USART_SendString(USART1,"demo");
```

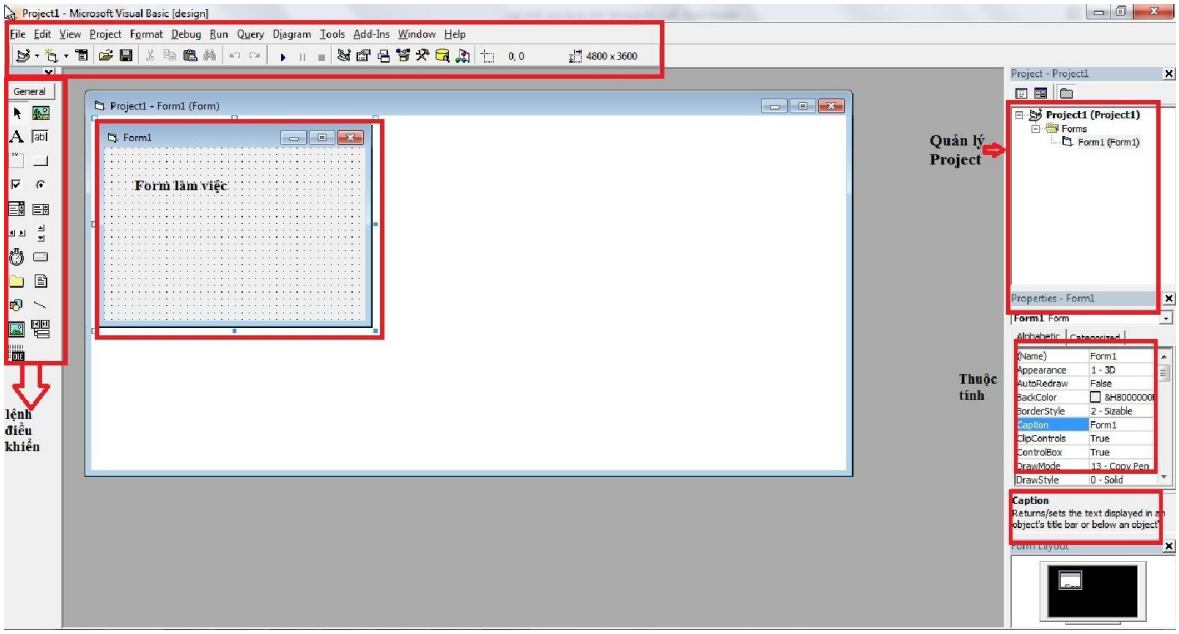
2. Lập trình giao diện truyền nhận dữ liệu UART dùng Visual Basic

- Khởi động chương trình:

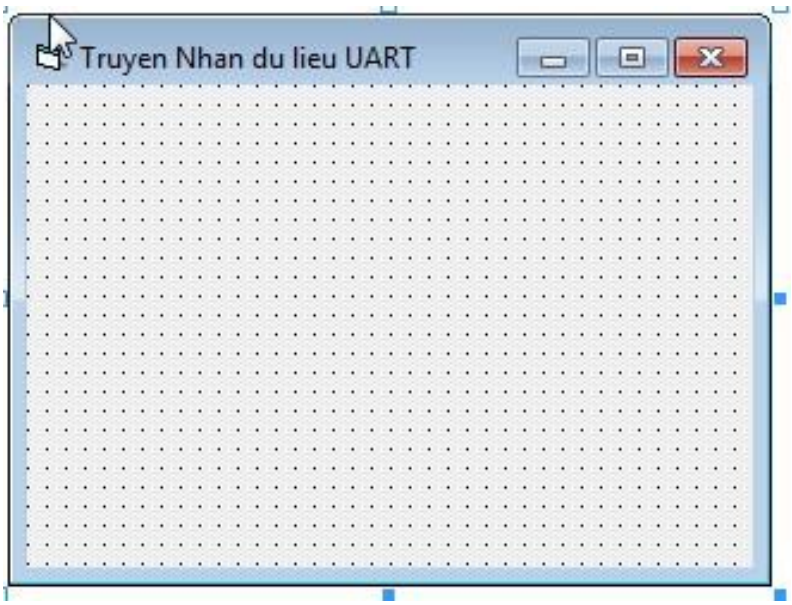


- Cửa sổ khởi động chọn **StandardEXE** để tạo một **form** mới

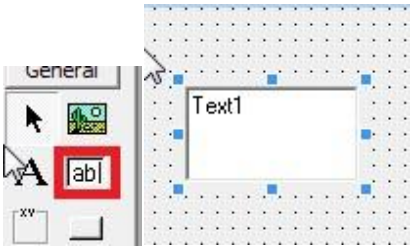
-Giao diện chương trình



- Click chuột vào **form1** để đổi tên cho **project** trong mục **caption**

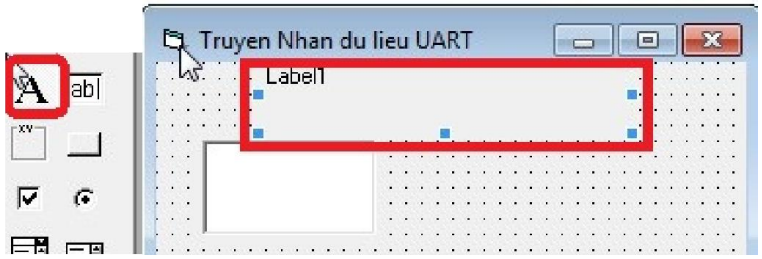


- Nhấp chuột vào biểu tượng textbox để lấy textbox ra màn hình

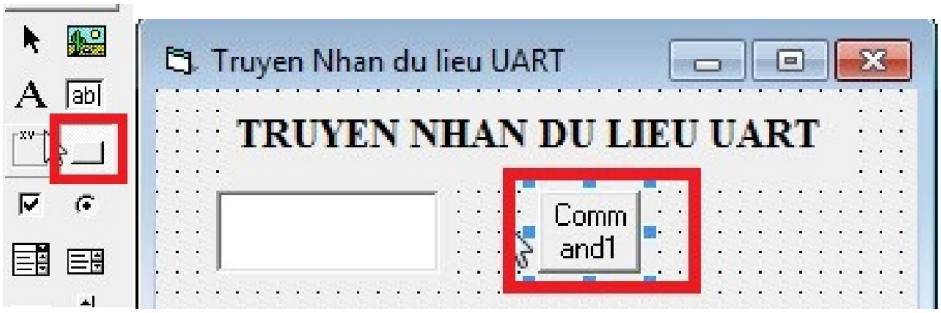


Trong phần thuộc tính của textbox xóa chữ text1 trong caption đi

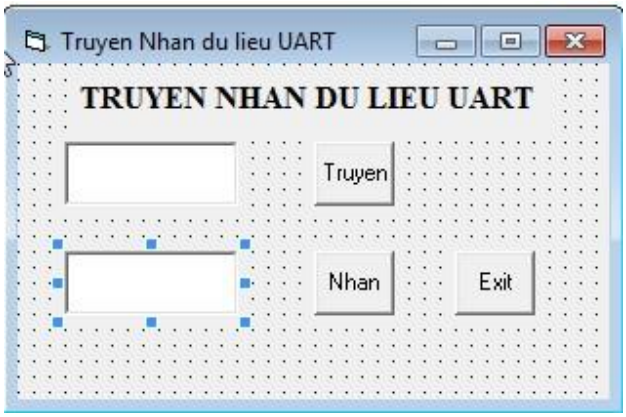
- Tiếp tục chọn label để kéo label vào chương trình, thay đổi tên label trong thuộc tính caption



- Tiếp tục với **button**



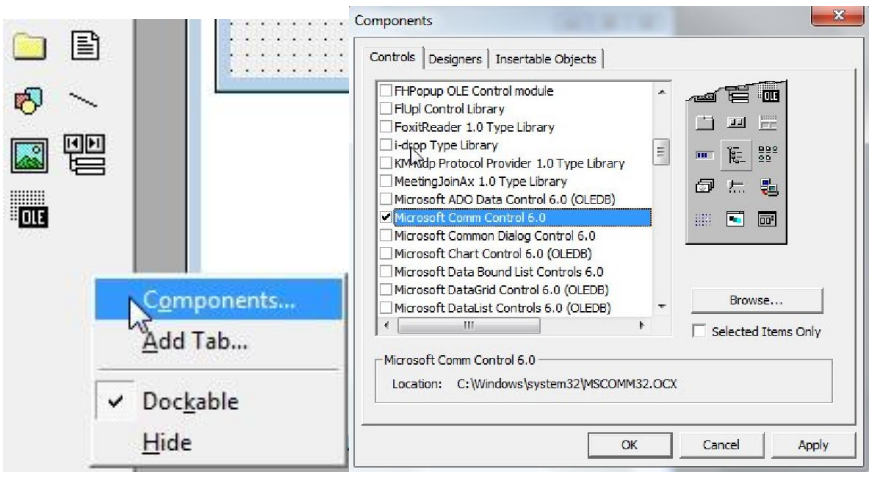
- Chỉnh sửa và thêm một số thành phần để có **form** như hình dưới



Như vậy ta đã tạo ra một **Form** các tham số **a,b** hiển thị với các **textbox1,2**. Nút truyền là **Command1** nút nhận là **Command2**, nút thoát là **Command3**

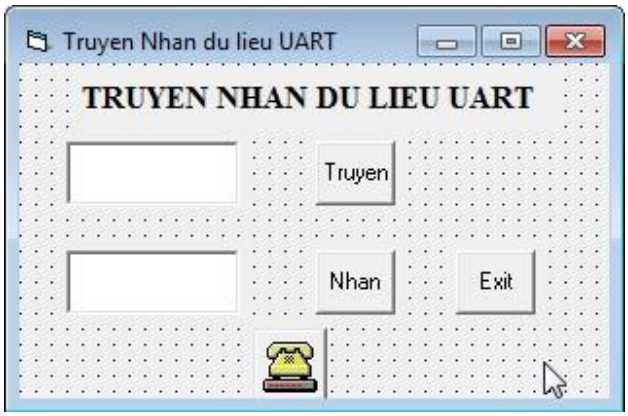
Form hoạt động như sau : Nhập các thông số trong **text1**, nhấn nút truyền để gửi dữ liệu trong **text1** ra cổng COM. Nhấn nút nhận thì dữ liệu nhận được sẽ hiển thị lên **text2**. Phím **Exit** để thoát khỏi chương trình

Vì Control để điều khiển cổng **COM-MSCOM** không phải control cơ bản nên nó không hiển thị trên tool, chúng ta phải lấy ra trong thư viện như sau

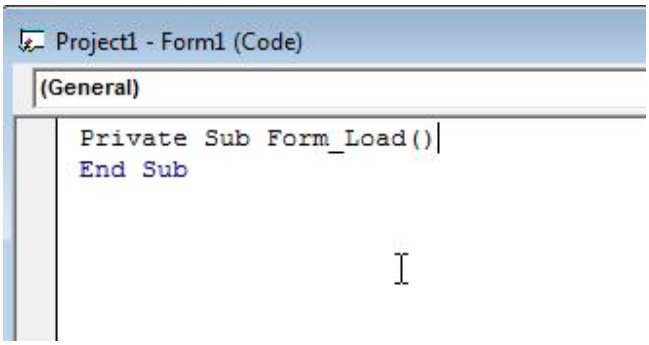


- Kéo thả **MSCOM** vào **form**

- Để soạn thảo code cho chương trình VB, ta click chuột vào vị trí bất kỳ trong form



- Giao diện chính



- Khai báo sử dụng cổng **COM**

```
Private Sub Form_Load()
MSComm1.CommPort = 5
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub
```

- Để viết mã cho nút **truyền** ta click vào nút truyền và code như sau. Khai báo thêm biến **s** để chứa giá trị của text

```
Dim s As String
Private Sub Command1_Click()
s = Text1.Text
MSComm1.Output = s
End Sub
```

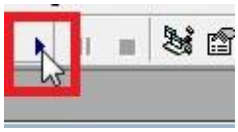
- Tương tự cho nút **nhận**

```
Private Sub Command2_Click()
s = MSComm1.Input
End Sub
```

- Và nút **Exit**

```
Private Sub Command3_Click()
End
End Sub
```

- Lưu **form** vừa tạo, chọn **Run** để chạy chương trình



- Kết quả :

- Chọn **file** – **Make tut.exe** để tạo file thực thi và chạy như một phần mềm thông thường