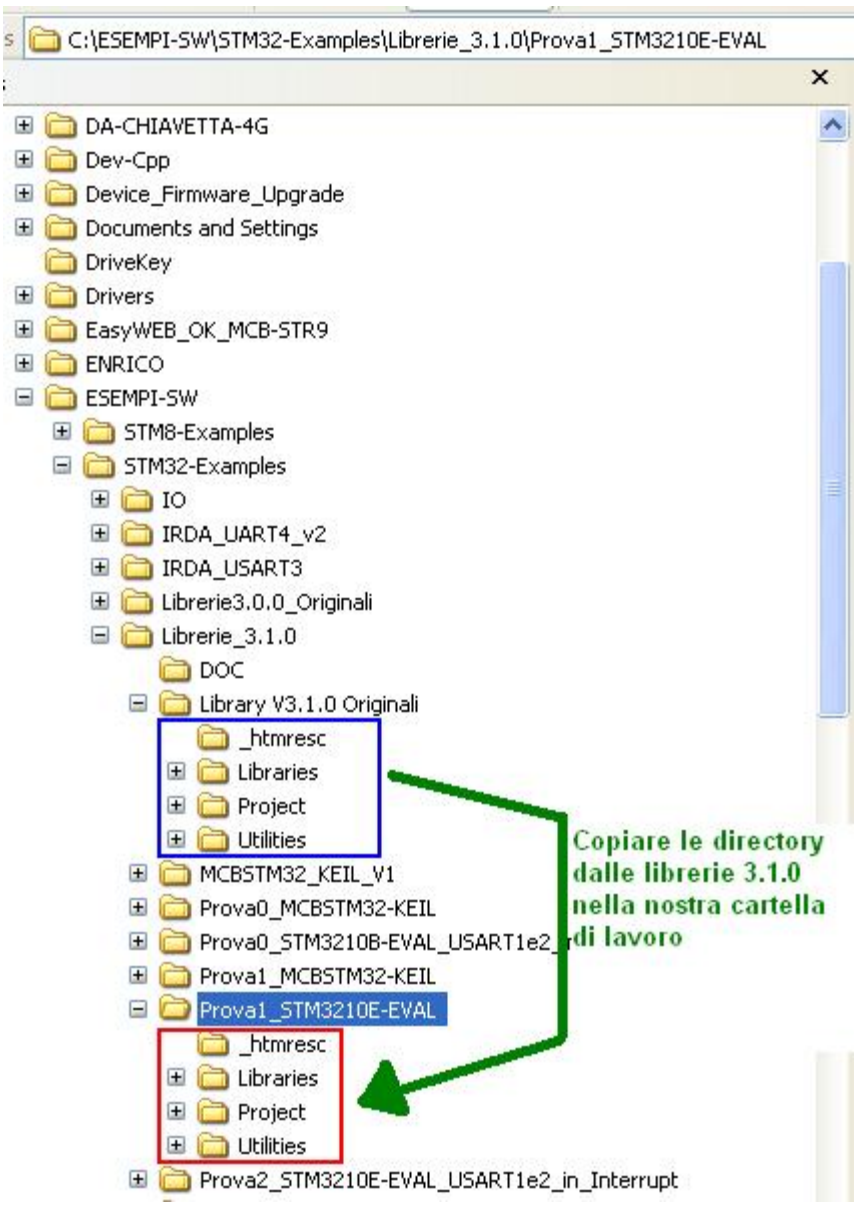
Usare le librerie 3.1.0 su STM32

By E.M.

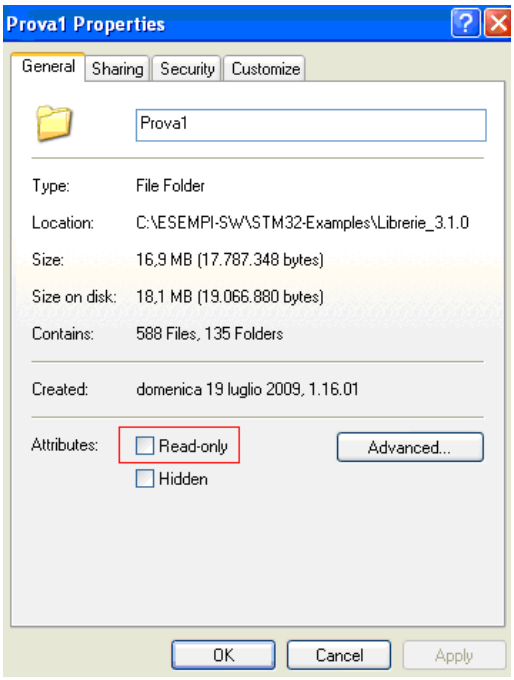
Per sviluppare un nuovo progetto, usando le librerie 3.1.0, la cosa più semplice da fare è partire dagli esempi forniti da STM che contengono già le configurazioni per gli IDE di KEIL, IAR e RAISONANCE.
Per capire come fare, di seguito, ci sono tre esempi sviluppati su due evaluation-board una di STM e l'altra di KEIL
Per tutti gli esempi si è usato l'ambiente di sviluppo di KEIL.

ESEMPIO basato su STM3210E-EVAL di STM

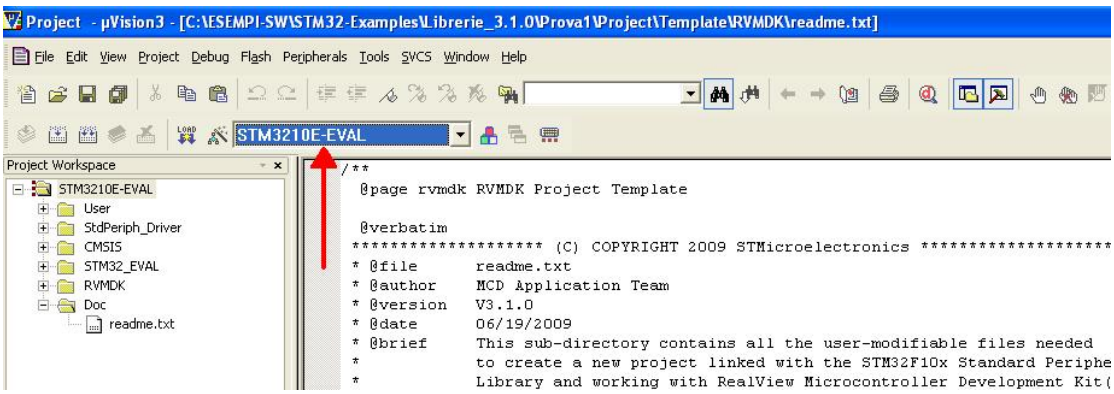
Per prima cosa creiamo una cartella di lavoro che chiameremo:
Prova1_STM3210E-EVAL
nel mio caso questa cartella è stata creata qui sotto:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\Prova1_STM3210E-EVAL
Adesso copiamo dalla directory in cui abbiamo scompattato le **librerie 3.1.0** le directory:
_htmresc
Libraries
Project
Utilities
nella nostra directory di lavoro **Prova1_STM3210E-EVAL**, al termine della copia dovremmo avere quanto riportato sotto nel rettangolo rosso.



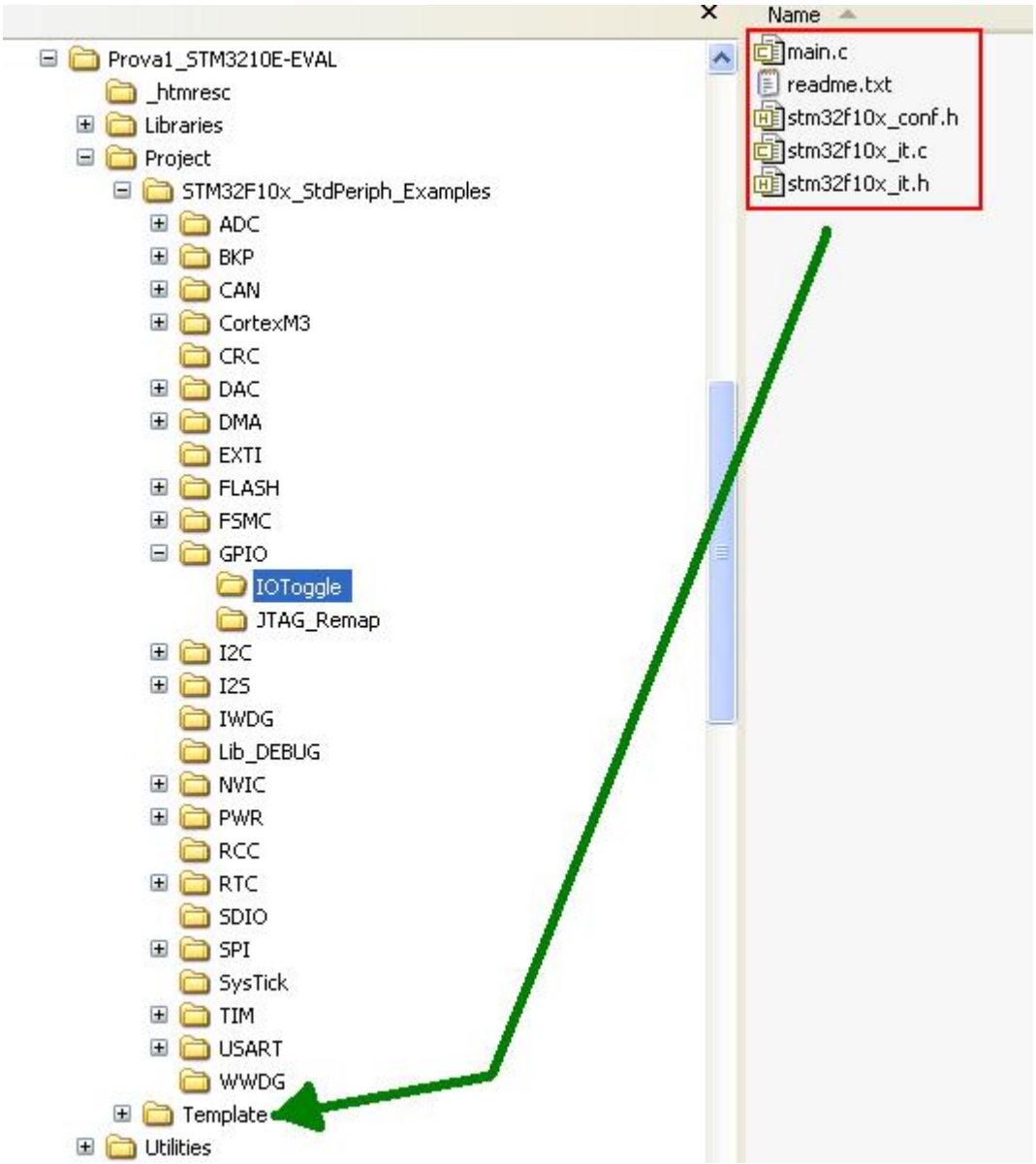
Controllate le **proprietà della directory** di lavoro (**Prova1_STM3210E-EVAL**) e **assicuratevi che NON sia "fleggato" Read-only**.
Se la directory fosse Raed-only togliete il flag e premete APPLY così da ottenere la schermata sotto riportata.



A questo punto andiamo nella directory sotto riportata:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\Prova1_STM3210E-EVAL\Project\Template\RVMDK
apriamo il file **Project.Uv2** cliccandoci due volte sopra.
Da **uVision3** selezioniamo il nome **STM3210E-EVAL** nella barra sotto evidenziata (freccia rossa).

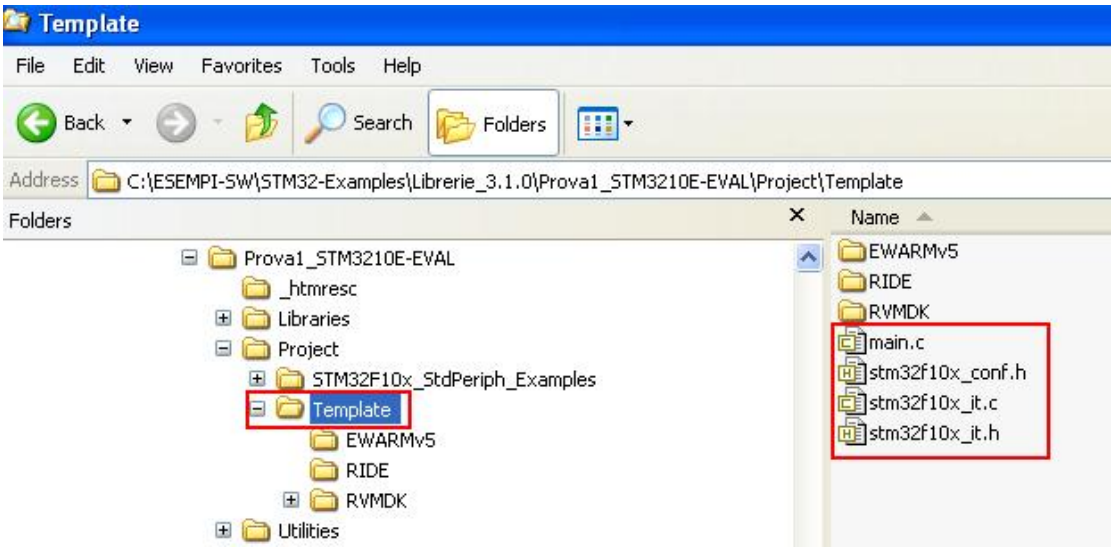


Ora, come prima prova, supponiamo di voler usare l'esempio di STM inerente il **IOToggle** che si trova nella directory sotto evidenziata in **blu**:

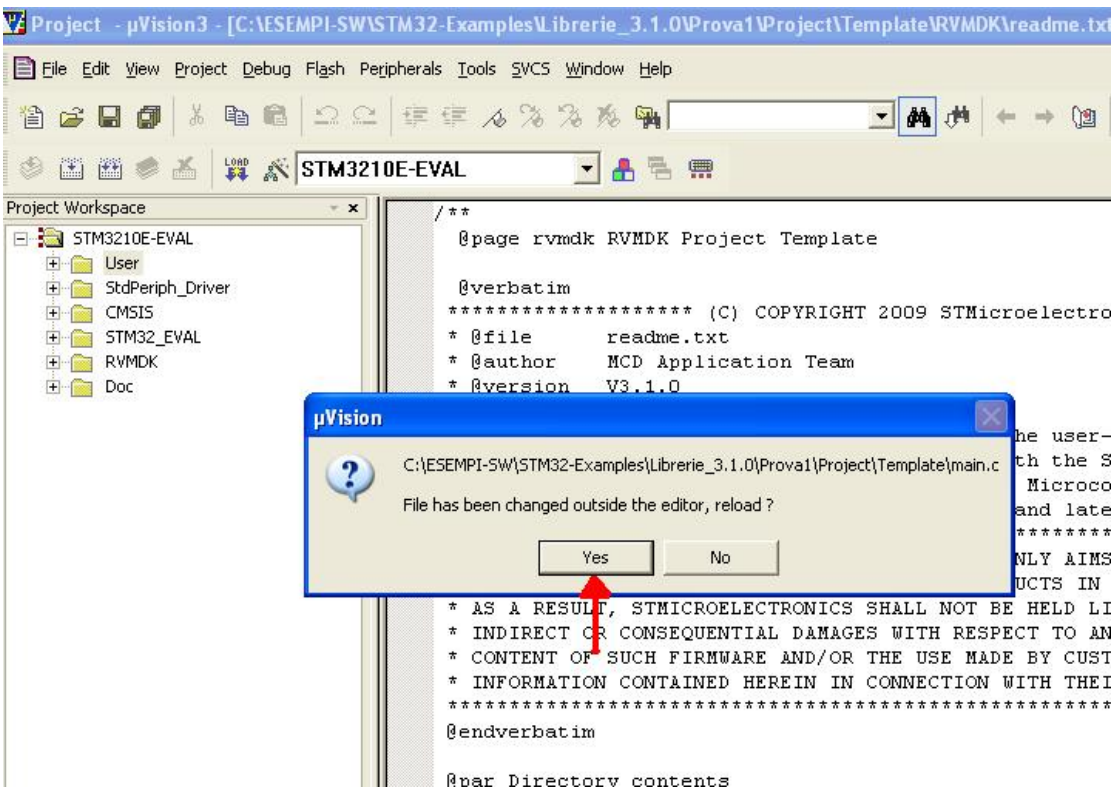


Per usare l'esempio **IOToggle** copiamo i files evidenziati sopra nel rettangolo rosso nella directory **Template**:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\ Prova1_STM3210E-EVAL\Project\Template

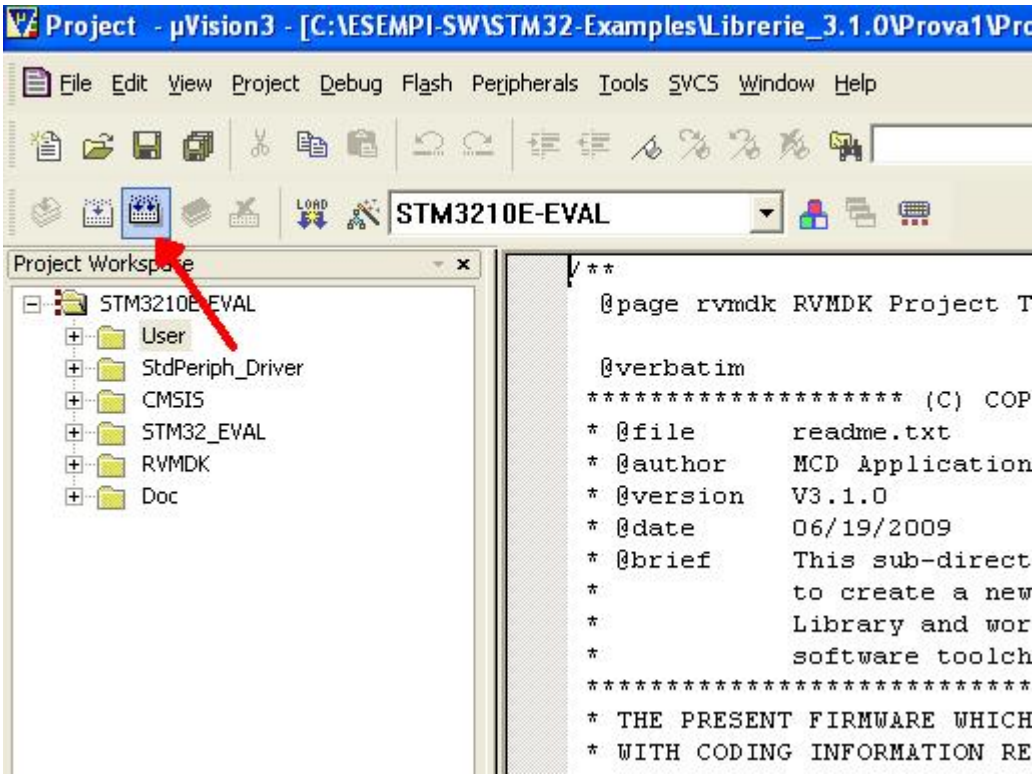
In pratica si vanno a sovrascrivere i files che erano già presenti nella directory **Template**, vedere sotto.



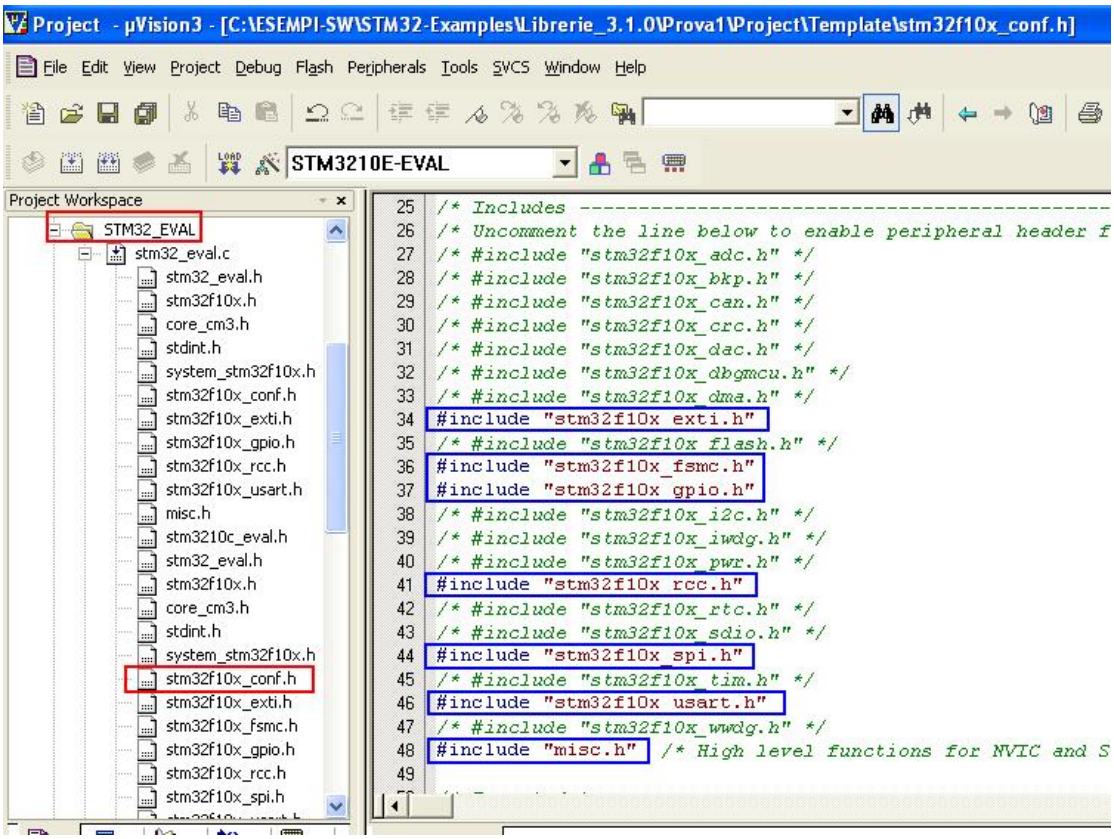
Torniamo all'ambiente uVision3 e dovremo trovare la finestra sotto riportata sotto che ci avvisa che alcuni files sono cambiati e che ci chiede se vogliamo ricaricarli, rispondiamo premendo**Yes** (freccia rossa sotto).



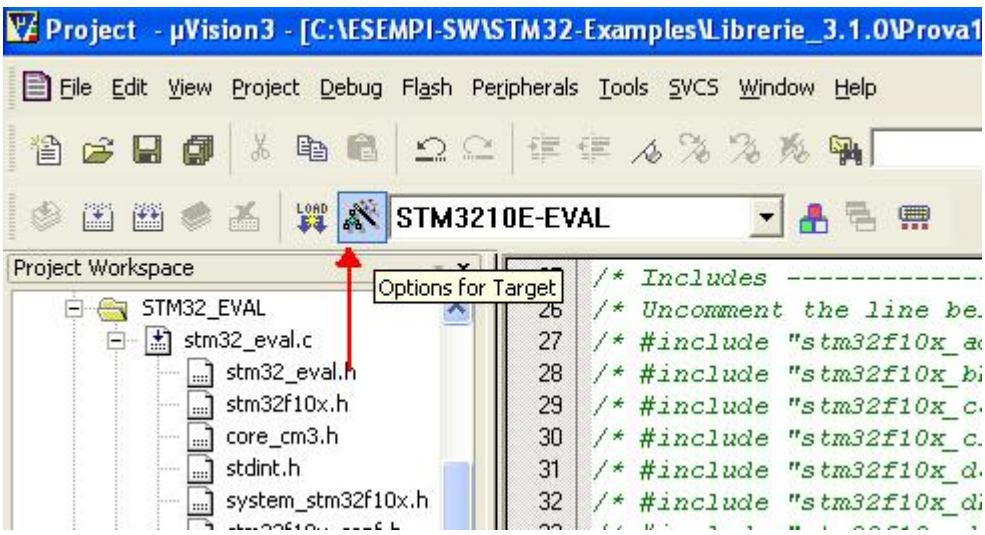
Fatti questi passaggi siamo pronti per compilare il nostro primo programma. Per compilare il programma premiamo sull'icona evidenziata dalla freccia rossa sotto riportata.



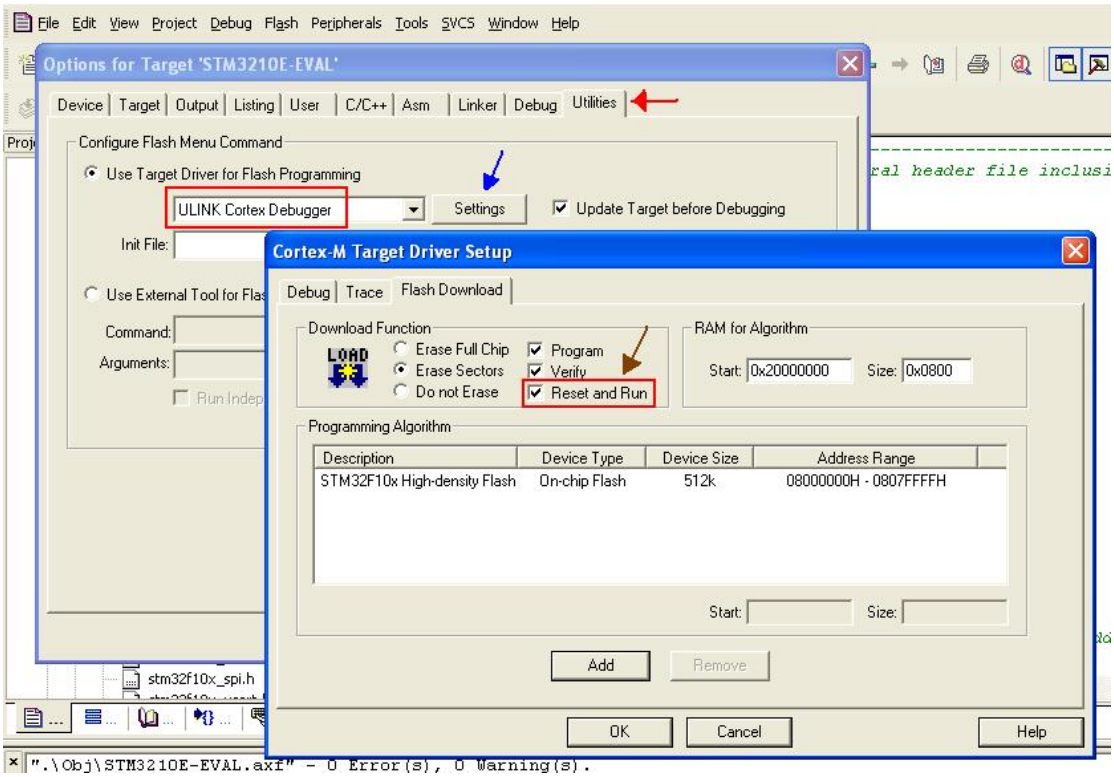
*La compilazione deve dare esito positivo ma… alcune volte capita che gli esempi, essendo stati convertiti dalla vecchia versione di librerie in automatico non sono ancora tutti provati e si può avere un certo numero di errori dovuti al semplice fatto che non sono stati inclusi tutti i file **h** che servono*.
Nel caso la compilazione vi avesse dato degli errori, aprite il file **stm32f10x_conf.h** che si trova nella directory **STM32_EVAL** (area Project Workspace) e verificate che tutte le linee cerchiate in **blu** siano senza commenti come sotto riportato.
Rifate la compilazione e questa volta tutto deve andare a buon fine.



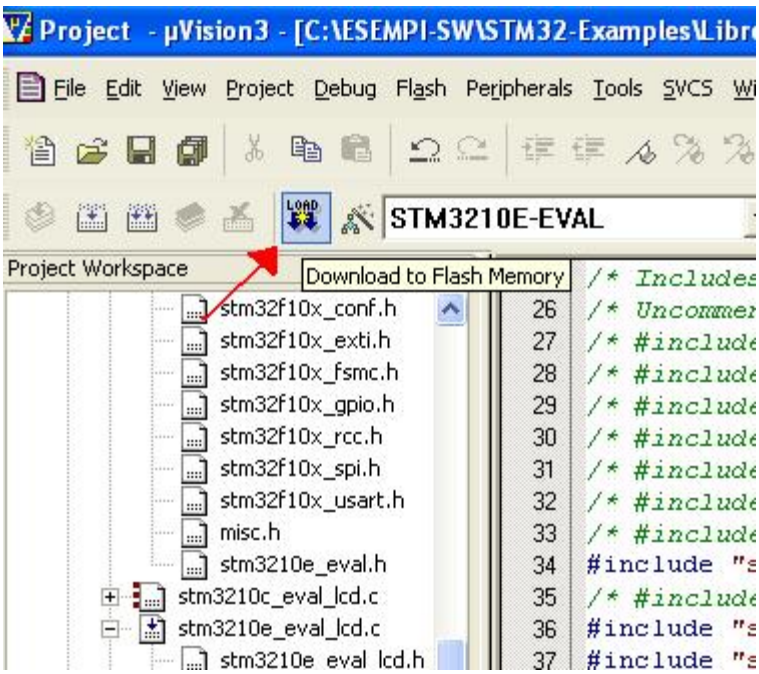
Adesso siamo pronti per scaricare sulla **STM3210E-EVAL** il programma per verificare se funziona tutto ma prima di fare ciò premete sull'icona **Option for Terget** come evidenziato sotto dalla freccia rossa.



Dalla finestra che compare (vedere sotto) premete su **Utilities** (freccia **rossa**) e poi premete su **Settings** (freccia **blu**) e nella nuova finestra “fleggate**”** **Reset and Run** (freccia **marrone**).
Fatto ciò date OK sulle finestre sino a chiuderle tutte.



Per scaricare e mandare in esecuzione il programma premete sul tasto evidenziato sotto e vedrete i led della STM3210E-EVAL lampeggiare.



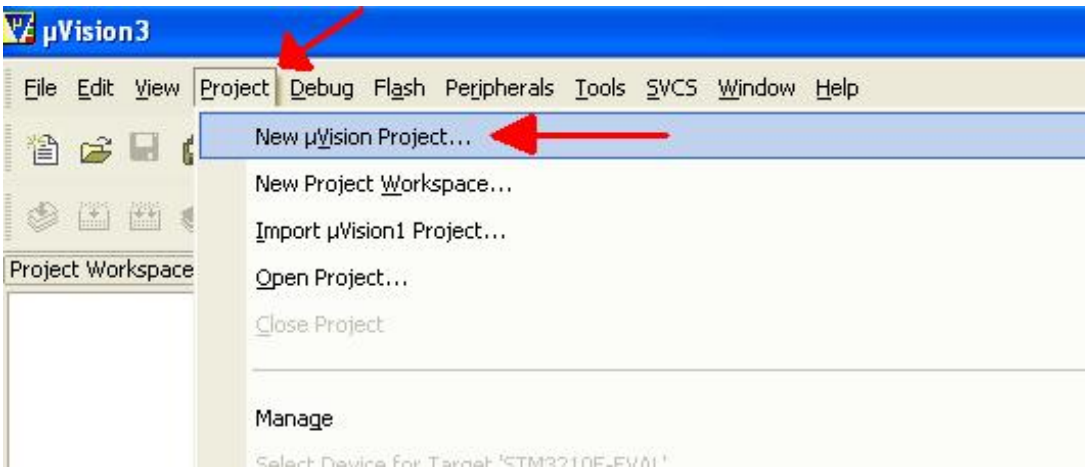
ESEMPIO basato su MCBSTM32 di KEIL partendo creando un nuovo progetto da zero

Questa è la strada più complicata per cui se volete seguire una via più semplice andate a pg.29

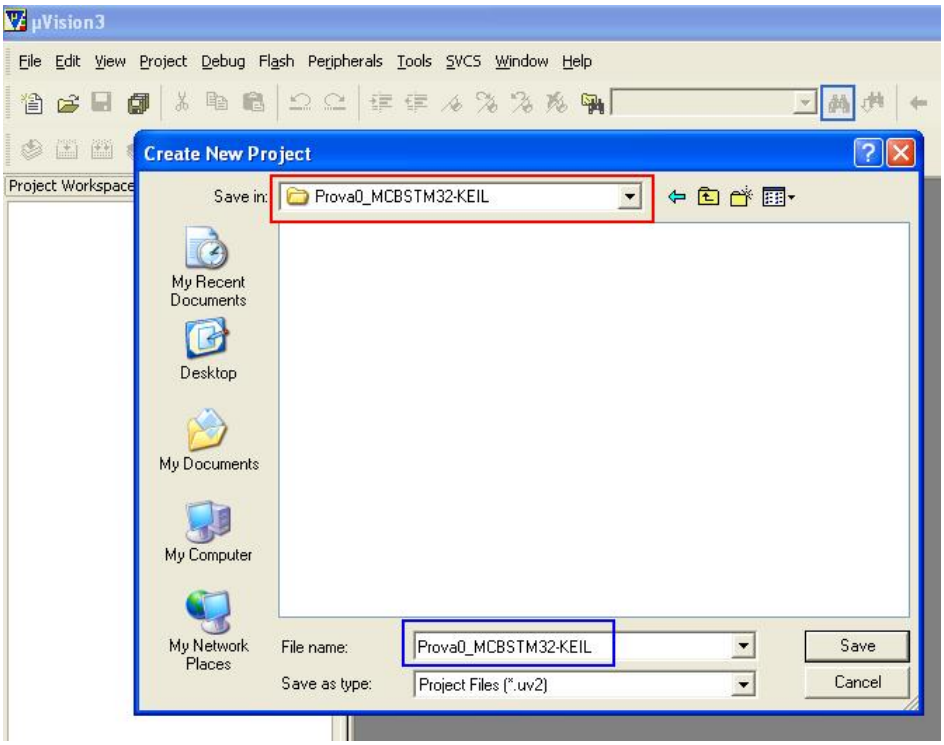
Supponiamo di creare un nuovo progetto che chiameremo **Prova0_MCBSTM32-KEIL** nella directory:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\ Prova0_MCBSTM32-KEIL
Supponiamo anche di avere le librerie per pilotare il **LCD** montato sulla **MCBSTM32-KEIL** che normalmente sono fornite da KEIL e che si chiamano: **lcd.c** e **lcd.h**
Supponiamo anche di voler far lampeggiare i **LED** sempre sfruttando gli esempi di STM.

Per prima cosa creiamo la directory sopra menzionata (**Prova0_MCBSTM32-KEIL**) e poi lanciamo il **Keil uVision3** cliccando sull'icona sotto riportata che dovreste avere sul vostro desk-top.

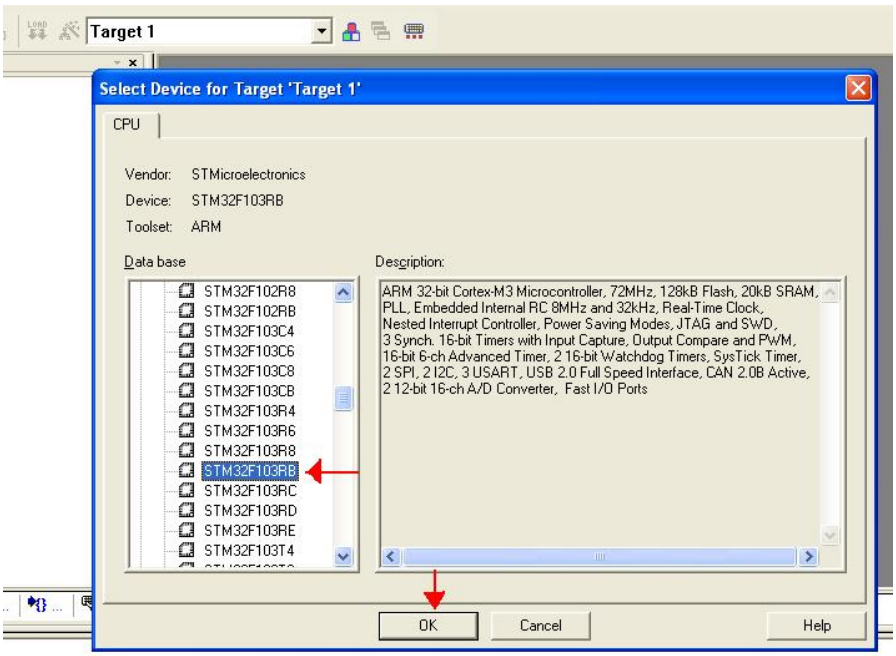
Nella finestra che compare cliccate su **Project** e poi su **New uVision Project** come evidenziato dalle frecce sotto.



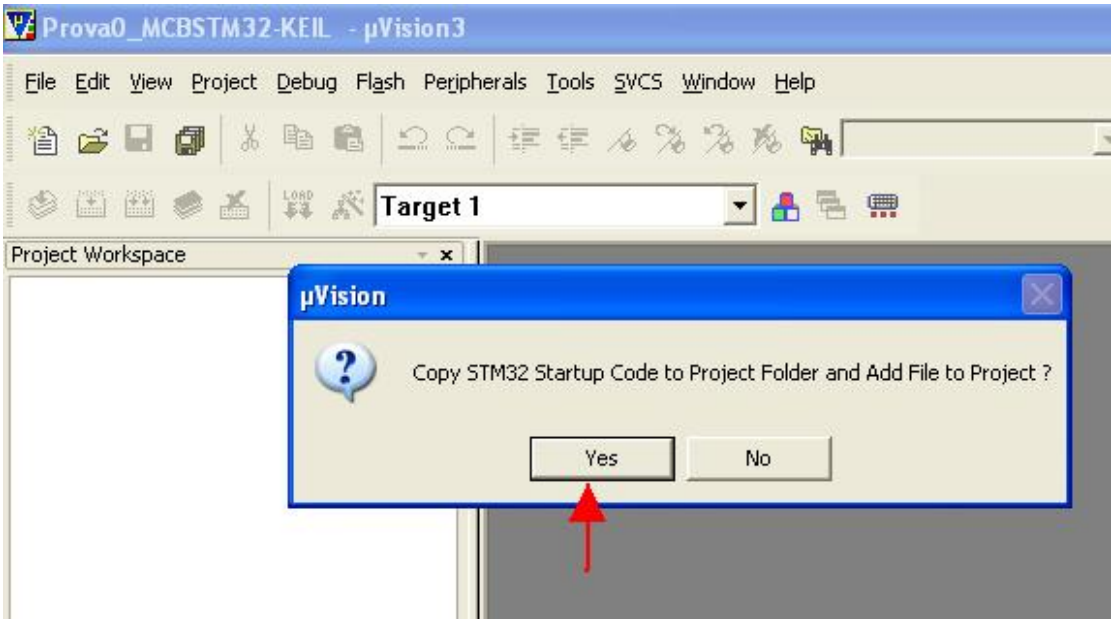
Nella nuova finestra selezioniamo la directory di lavoro che abbiamo creato in precedenza (rettangolo rosso) e poi diamo un nome al nostro progetto che chiameremo ancora **Prova0_MCBSTM32-KEIL** (rettangolo blu) e poi premiamo **Save** (vedere l'immagine sotto riportata).



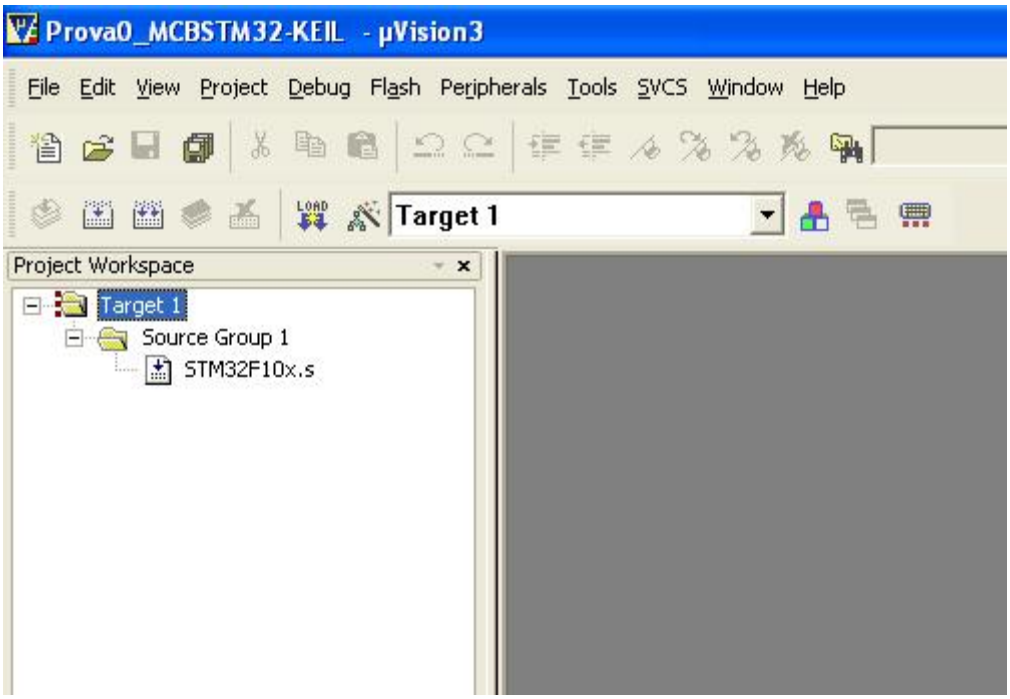
A questo punto ci si aprirà una nuova finestra che ci permetterà di scegliere la MCU con cui vogliamo lavorare.
Da questa finestra selezioniamo **ST** e poi **STM32F103RB** che è la MCU montata sulla **MCBSTM32-KEIL** e poi premiamo OK (vedere immagine sotto).



Adesso vi comparirà la finestra sotto riportata che vi chiede se volete copiare lo **StartUp Code file** e aggiungerlo nel vostro progetto.
Rispondete **Yes** come evidenziato sotto dalla freccia.

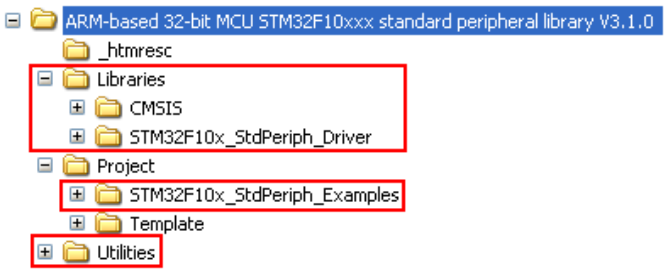


Il progetto è stato creato e dovrete avere la struttura sotto riportata dove, l'unico file che troverete è lo StartUp file (**STM32F10x.s**).



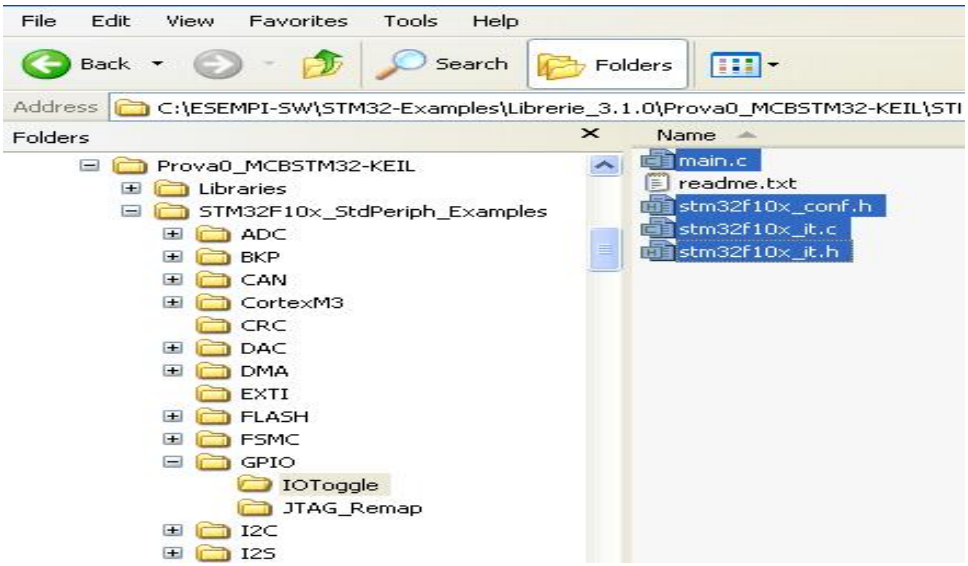
Adesso **chiudiamo il uVision3** perché dobbiamo copiare nella directory di lavoro (**Prova0_MCBSTM32-KEIL**) i files e le directory che ci servono.

Visto che vogliamo usare **le librerie di STM ver.3.1.0** copiamo nella nostra directory di lavoro (**Prova0_MCBSTM32-KEIL**) le directory della libreria di STM qui sotto riportate:
Libraries
Utilities
STM32F10x_StdPeriph_Examples
Queste directory si prendono dalla directory in cui abbiamo scompattato le librerie 3.1.0 di STM, sotto è riportato l'albero della mia libreria originale con cerchiati in rosso le directory sopra menzionate.



Abbiamo anche detto che vogliamo visualizzare delle scritte sul LCD per cui copiamo, sempre nella nostra directory di lavoro, i files:
lcd.c
lcd.h
Questi due files si trovano allegati agli esempi forniti da KEIL per le sue demo board.

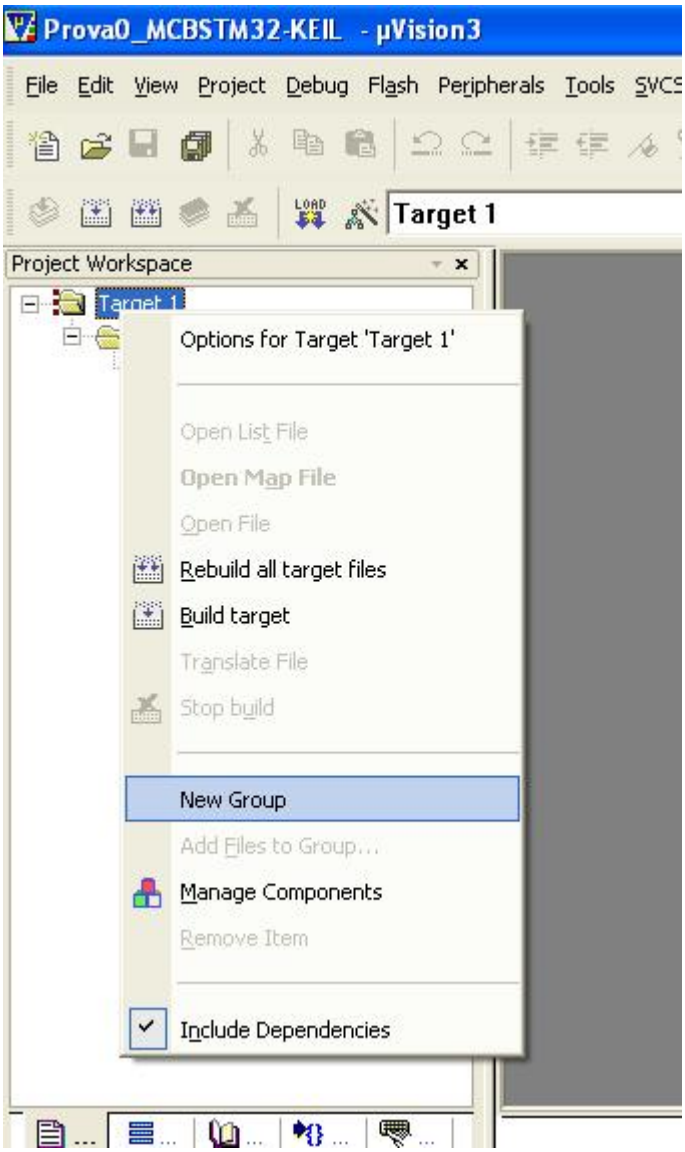
Se vi ricordate, abbiamo anche detto che vogliamo far lampeggiare i LED della **MCBSTM32-KEIL** e per fare ciò useremo l'esempio fornito da STM che prendiamo da:
…/ STM32F10x_StdPeriph_Examples/GPIO/IOToggle
Da questa directory copiamo i files, evidenziati sotto in blu, nella nostra directory di lavoro (**Prova0_MCBSTM32-KEIL**).



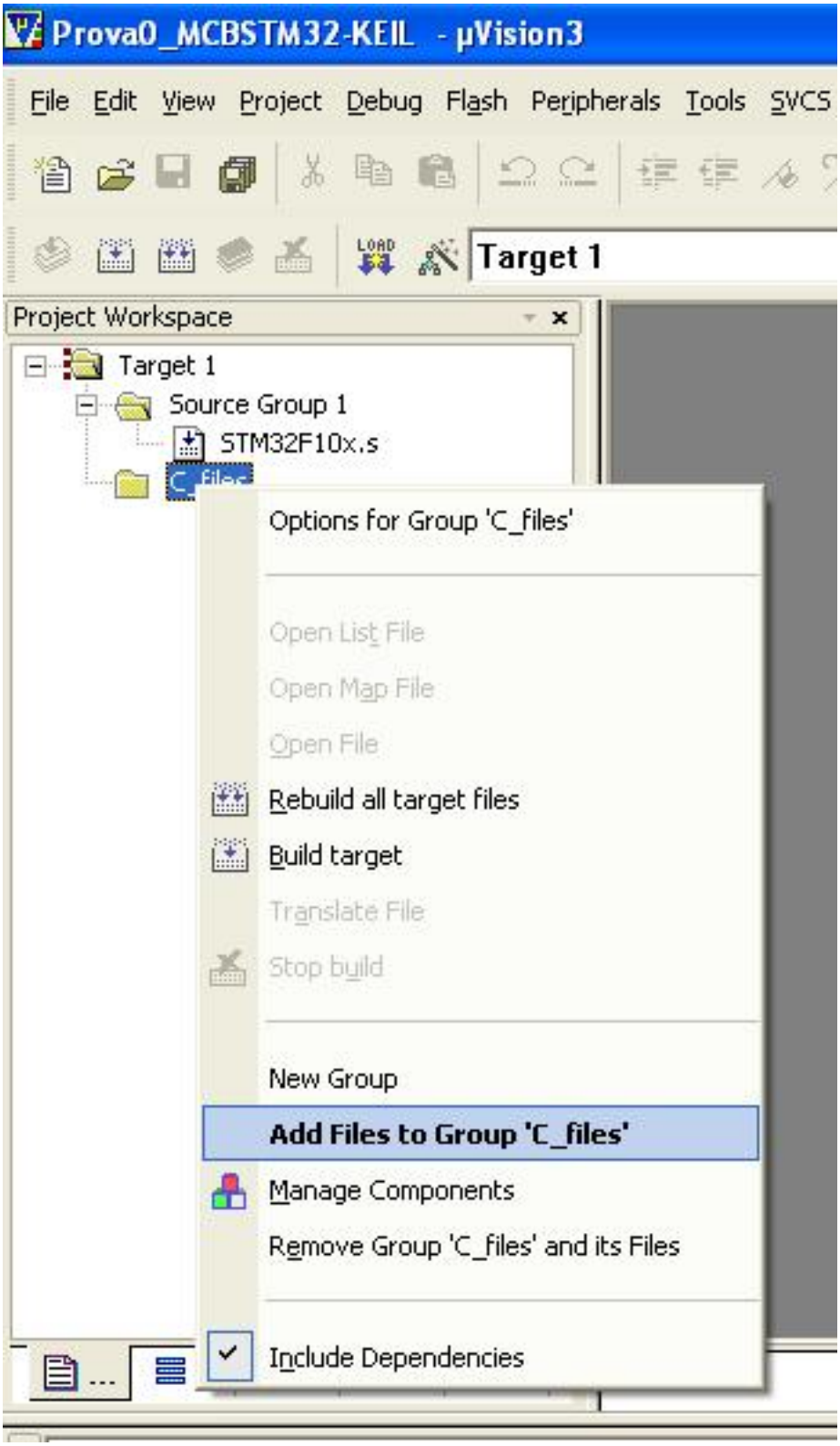
Arrivati a questo punto verifichiamo che i file presenti nella nostra directory non siano protetti e se lo fossero sproteggiamoli come spiegato a pg.3.

Adesso **riapriamo uVision3**.

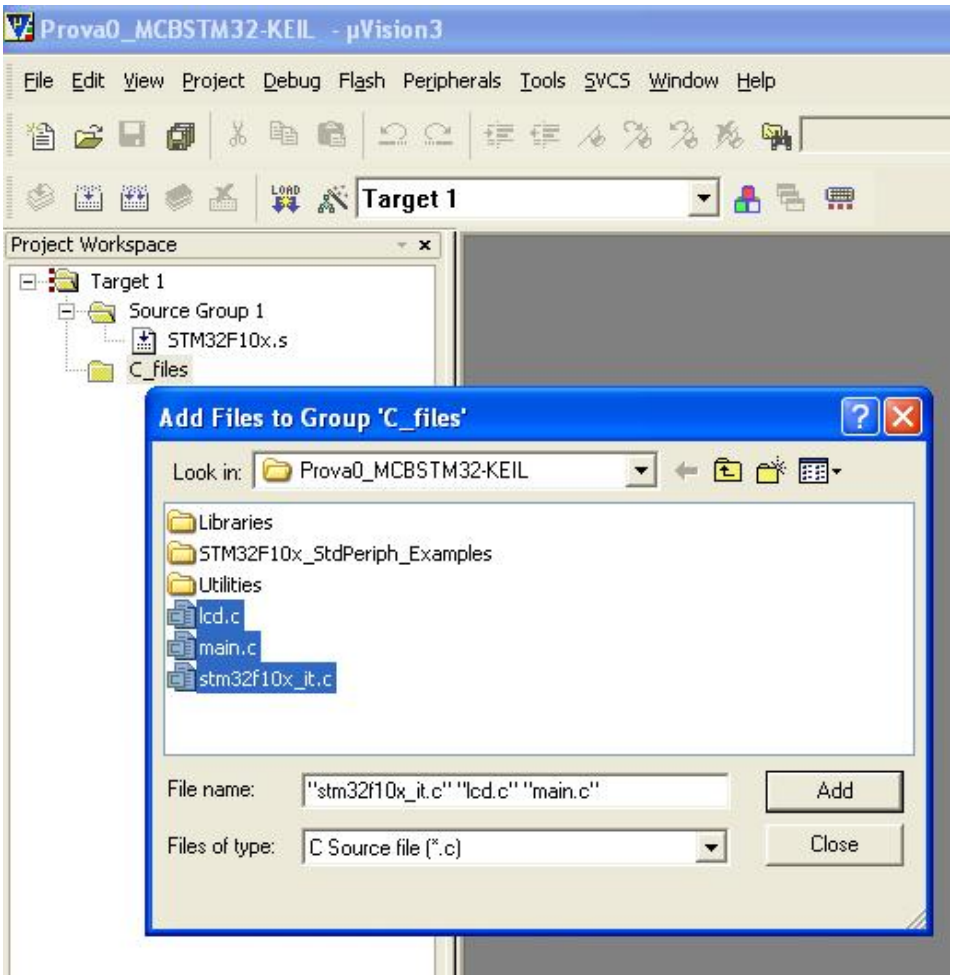
Evidenziamo la scritta **Target 1** (vedere sotto) e per mantenere una suddivisione logica dei files **creiamo la cartella C_files** premendo con **il tasto destro su Tagret 1** e dalla finestra che compare scegliamo **New Group** (vedere sotto).



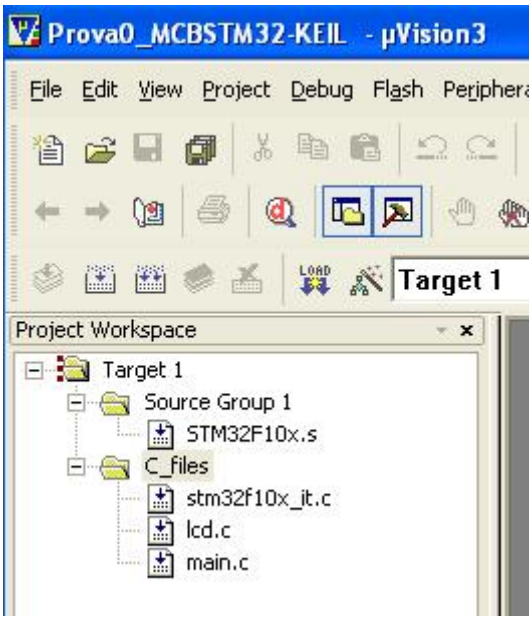
A questo punto ci comparirà una nuova directory a cui daremo il nome di **C_files**.
Nuovamente **evidenziamo** la directory **C_files** e nuovamente premiamo con il tasto destro del mouse e scegliamo **Add File to Group 'C_files'** come sotto riportato.



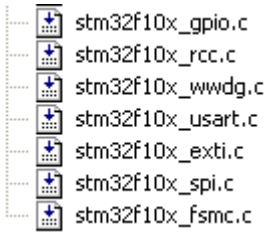
Dalla finestra che compare selezioniamo i files in blu, sotto evidenziati, premiamo **Add** e poi premiamo **Close**.



A questo punto se apriamo la directory **C_files** dobbiamo trovare i files sotto riportati.



Siamo a buon punto ma serve includere altri files e per fare ciò nuovamente evidenziamo **C_files** e poi con il tasto destro premiamo su **Add File to Group 'C_files'** e con la barra di navigazione andiamo nella directory:
…\Prova0_MCBSTM32-KEIL\Libraries\STM32F10x_StdPeriph_Driver\src
e includiamo i files sotto riportati.



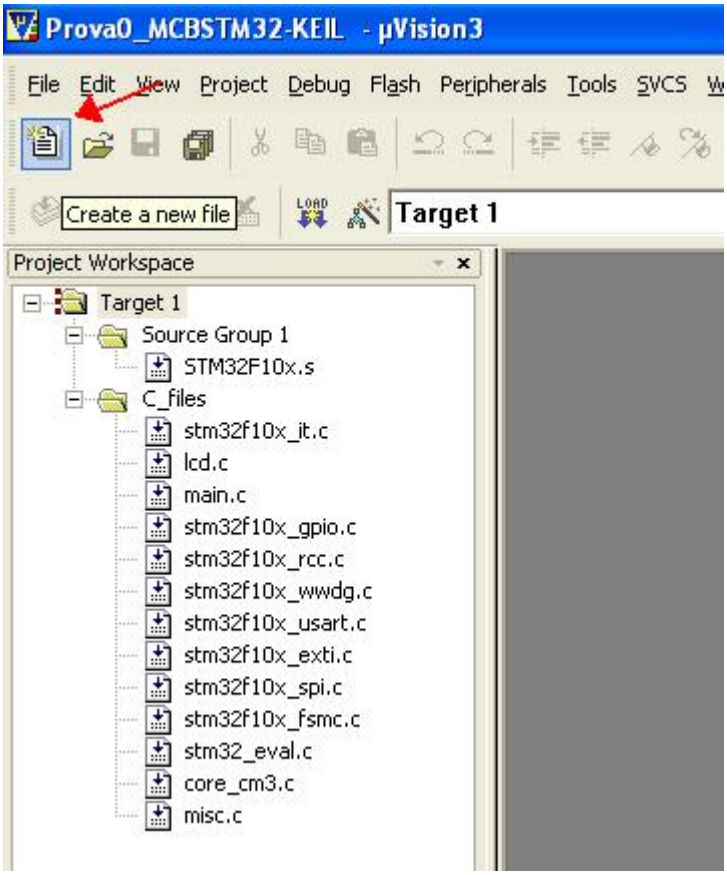
Adesso dobbiamo inserire il file **stm32_eval.c** che si trova nella directory:
…\Prova0_MCBSTM32-KEIL\Utilities\STM32_EVAL

Ancora dobbiamo inserire i files **core_cm3.c** e **system_stm32f10x.c** che si trova nella directory:
…\Prova0_MCBSTM32-KEIL\Libraries\CMSIS\Core\CM3

Abbiamo quasi finito le inclusioni, manca solo da includere **misc.c** che si trova nella directory:
…\Prova0_MCBSTM32-KEIL\Libraries\STM32F10x_StdPeriph_Driver\src

Finalmente abbiamo terminato le inclusioni e per dirla tutta, abbiamo incluso anche dei files che al momento non ci servono ma che useremo in seguito per sviluppare ulteriormente il nostro programma.

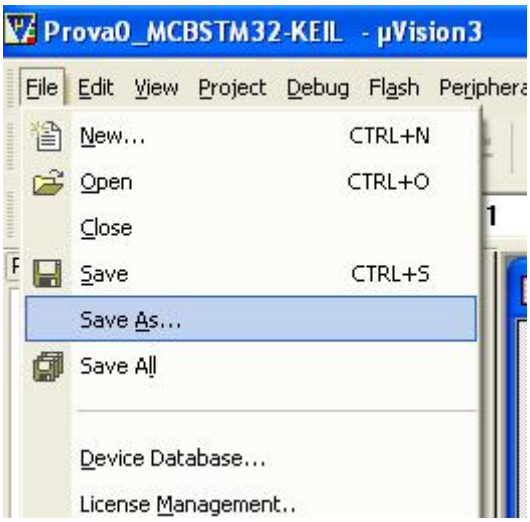
Adesso dobbiamo creare un file contenete le nostre definizioni che useremo nel nostro programma.
Per fare ciò **premiamo** sull'icona **Create a new file** sotto evidenziata.



Nel nuovo file scriviamo le definizioni qui sotto riportate:

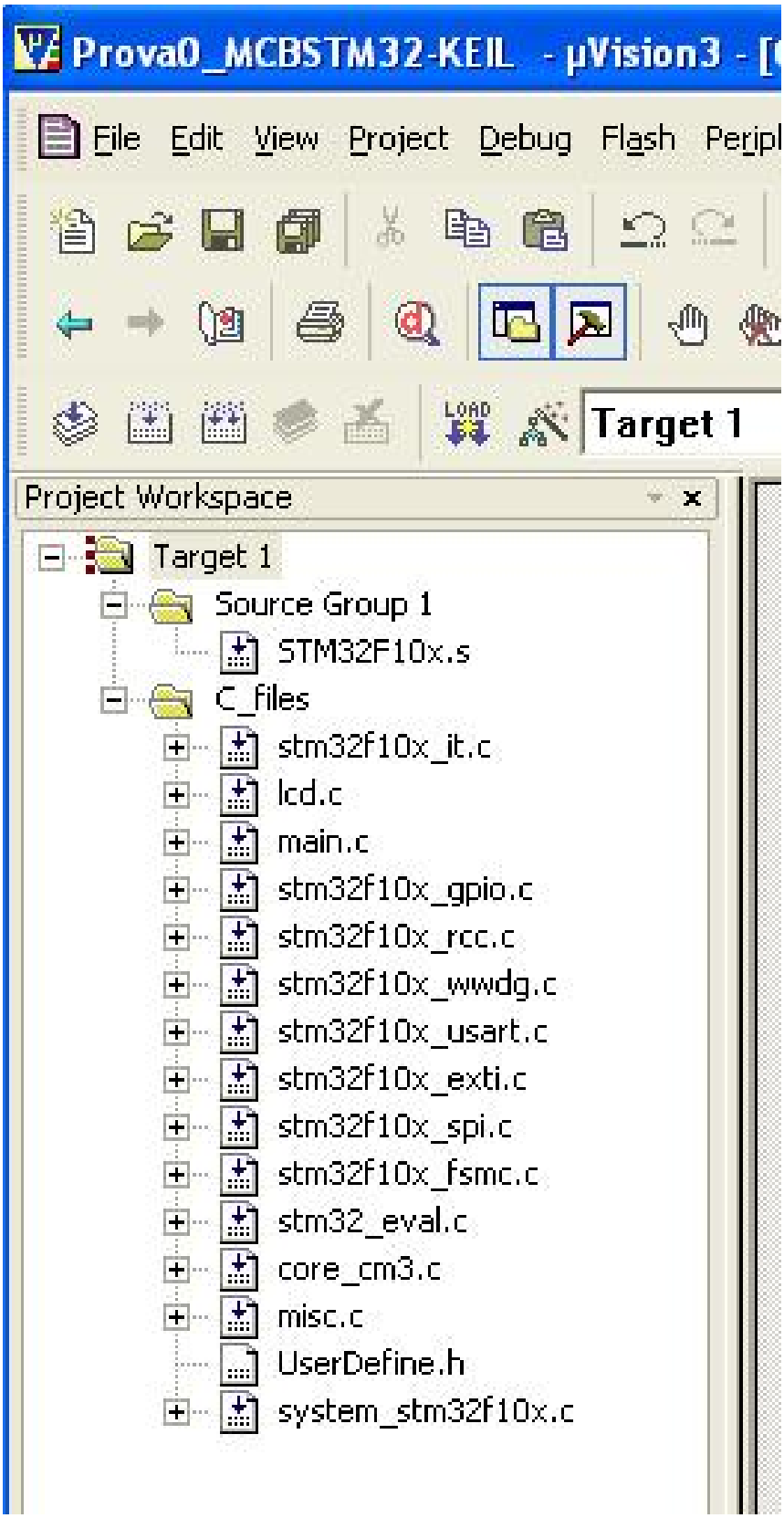
```
// Define for LEDs
#define LEDPort   GPIOB
#define LED8      GPIO_Pin_8
#define LED9      GPIO_Pin_9
#define LED10     GPIO_Pin_10
#define LED11     GPIO_Pin_11
#define LED12     GPIO_Pin_12
#define LED13     GPIO_Pin_13
#define LED14     GPIO_Pin_14
#define LED15     GPIO_Pin_15
```

Fatto ciò salviamo il file usando l'opzione **File/Save As…** (sotto riportata) e diamogli il nome **UserDefine.h**



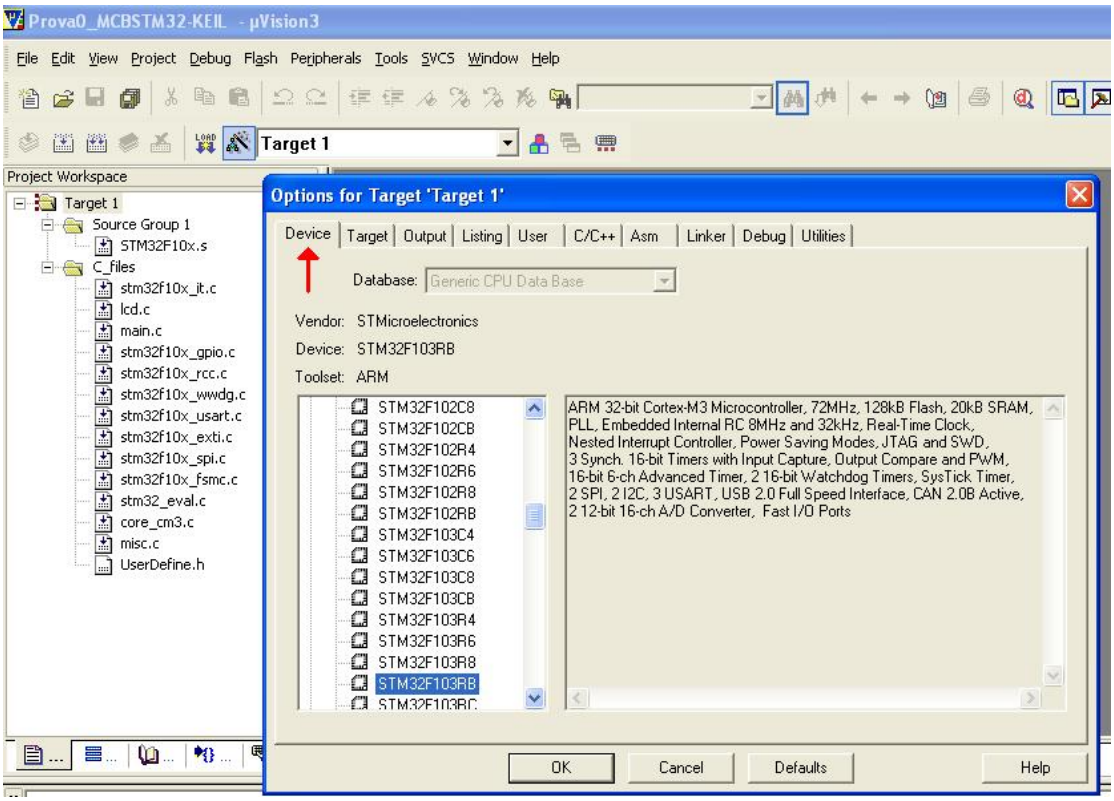
Nuovamente includiamo il file che abbiamo appena creato nel nostro progetto ripetendo le operazioni che abbiamo fatto in precedenza ovvero:
Evidenziamo la directory **C_files**, premiamo con il tasto destro del mouse e scegliamo **Add File sto Group 'C_files'** e inseriamo **UserDefine.h**

Al termine di questa fase dobbiamo avere la struttura sotto riportata.

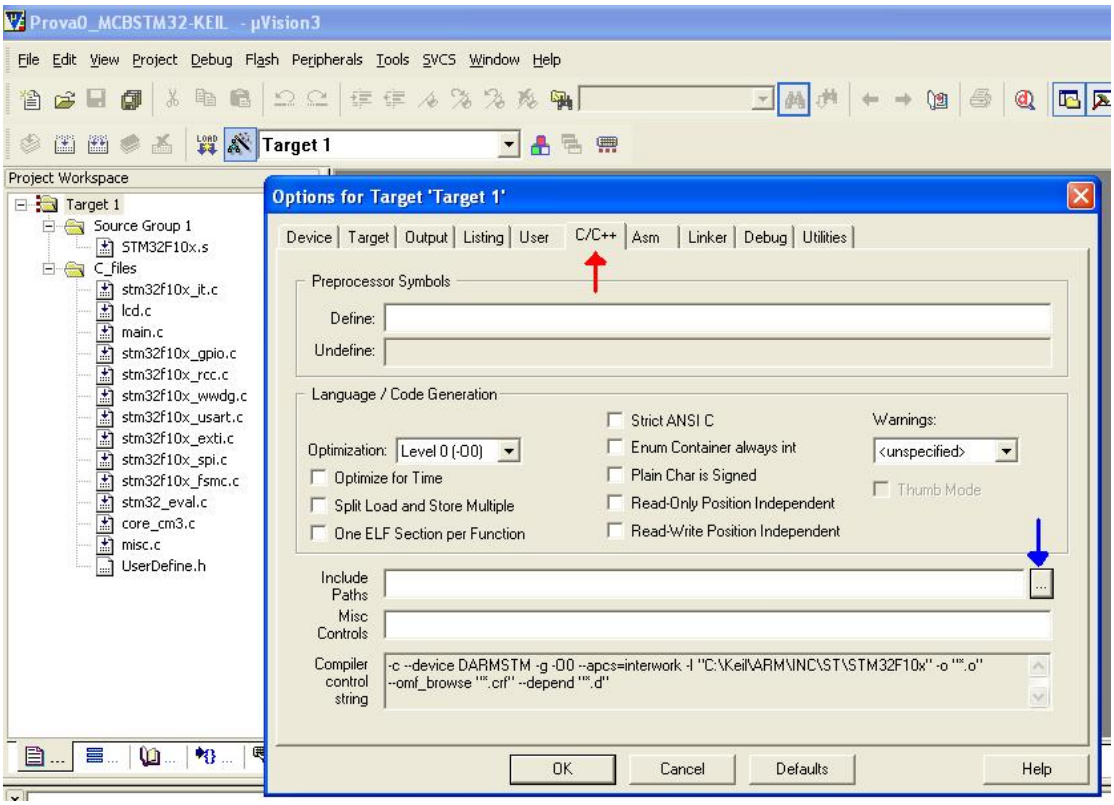


Arrivati a questo punto dobbiamo configurare l'ambiente di sviluppo e per fare ciò premiamo sull'icona riportata sotto chiamata **Options for Target**.

Dalla finestra che compare premete su **Device** e verificate che sia selezionata la mcu **STM32F103RB** come riportato sotto.

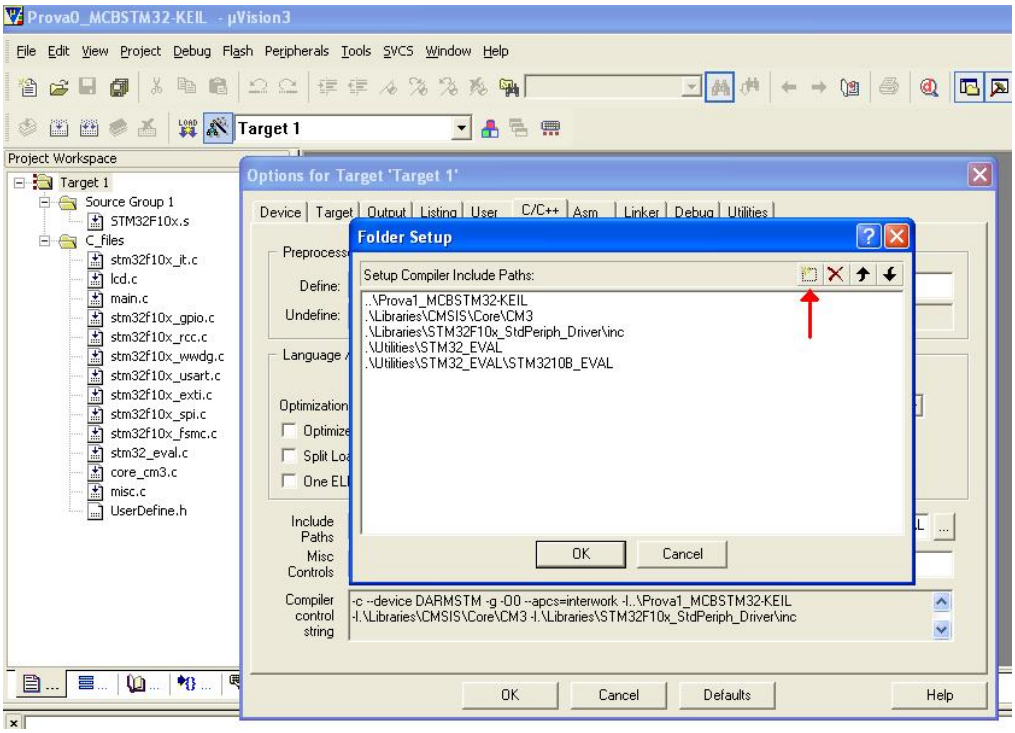


Adesso premete su **C/C++** come evidenziato sotto dalla freccia rossa.

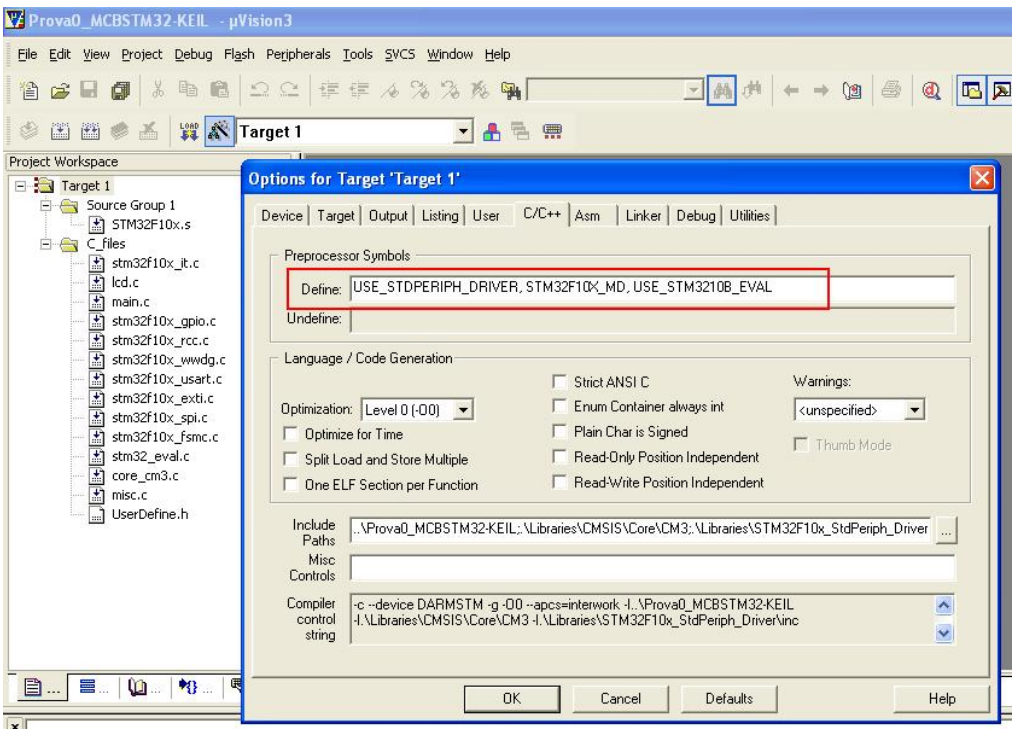


Qui dovremo includere le paths che il compilatore userà per cercare i files.
In pratica dovremo includere la linea sotto rispettando esattamente la sintassi riportata e senza aggiungere spazi:
..\Prova0_MCBSTM32-KEIL;.\Libraries\CMSIS\Core\CM3;.\Libraries\STM32F10x_StdPeriph_Driver\inc;.\Utilities\STM32_EVAL;.\Utilities\STM32_EVAL\STM3210B_EVAL
Il modo più veloce è ricopiare la riga sopra e incollarla nella riga **Include Paths**.
Un altro modo per inserire le path è premere sul rettangolo con i puntini alla fine della linea **Include Paths** (vedere sopra la freccia blu).

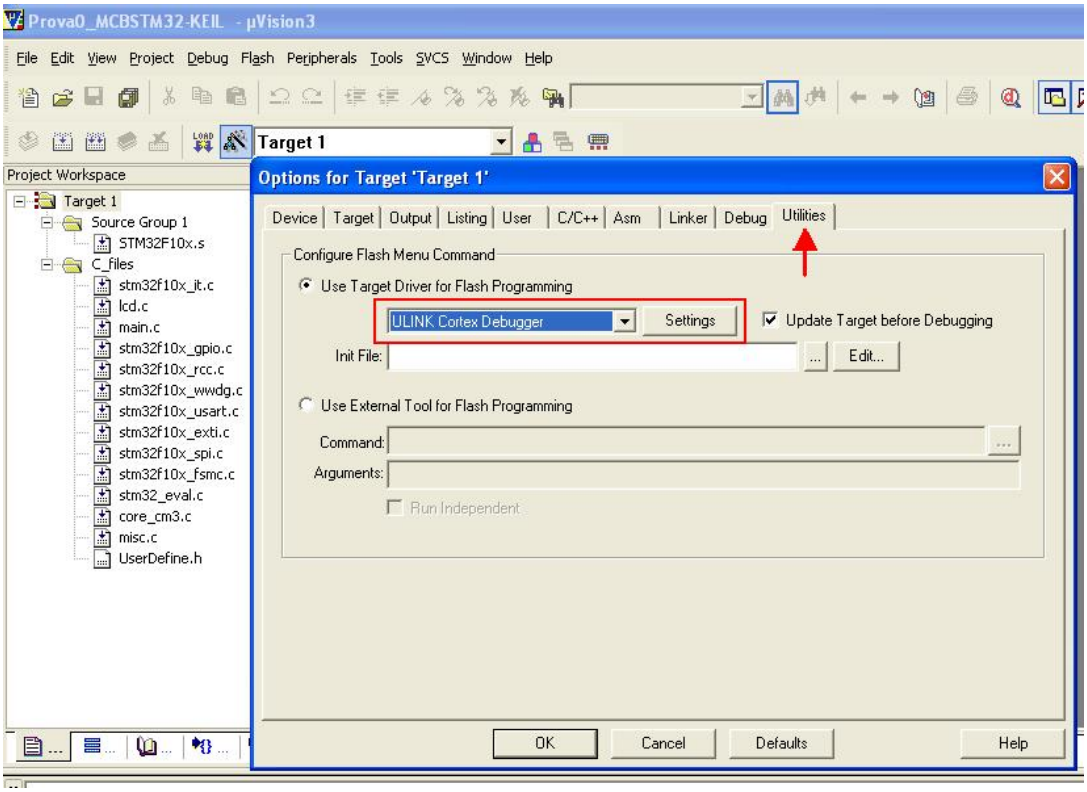
Se scegliamo questa seconda strada dalla finestra che si appare dobbiamo premere sull'icona evidenziata dalla freccia rossa (vedere sotto) e poi specificare una ad una le directory come sotto riportate.
In pratica si inseriscono le cinque linee sotto e poi si preme OK.



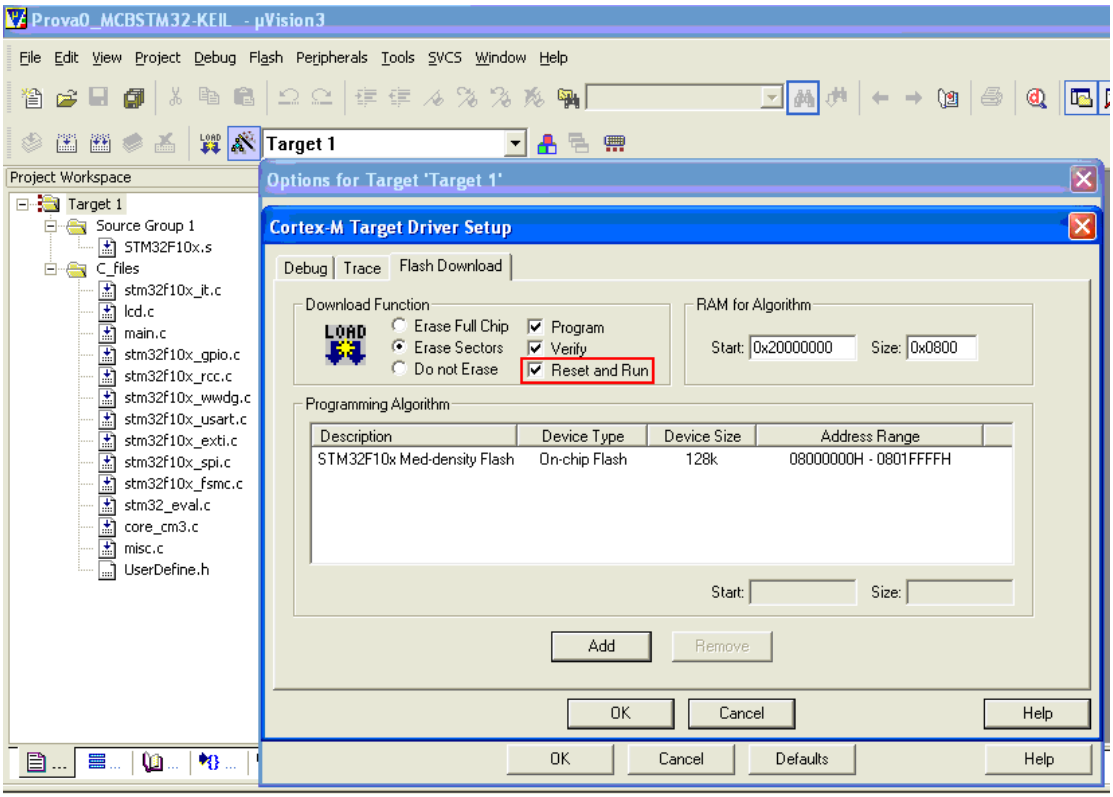
Adesso, in corrispondenza della linea **Define**, dobbiamo inserire la scritta **USE_STDPERIPH_DRIVER, STM32F10X_MD, USE_STM3210B_EVAL** come sotto evidenziato nel rettangolo rosso. Questo ci serve per usare le definizioni fatte da STM.



A questo punto premiamo su **Utilities** e selezioniamo **ULINK Cortex Debuger** come evidenziato sotto nel rettangolo rosso.



L'ultimo passaggio è selezionare **Settings** e impostare la pagina come sotto.



Per confermare le selezione diamo **OK** due volte.

Adesso non ci resta che modificare i file:

main.c
stm32_eval.h
stm32f10x_conf.h

Le modifiche che apporteremo serviranno ad adattare il codice degli esempi di STM per farli funzionare sulla nostra MCSTM32-KEIL.
Di seguito è spiegato come fare.

Modifiche a **main.c**

Nella sezione **int main(void)** sino **a while (1)** sostituite quello che trovate scritto con quanto qui sotto riportato.

```
int main(void)
{
  /* System Clocks Configuration **********************************************/
  RCC_Configuration();

  /* Configure all unused GPIO port pins in Analog Input mode (floating input
     trigger OFF), this will reduce the power consumption and increase the device
     immunity against EMI/EMC *************************************************/
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                         RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD |
                         RCC_APB2Periph_GPIOE, ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  GPIO_Init(GPIOC, &GPIO_InitStructure);
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  GPIO_Init(GPIOE, &GPIO_InitStructure);

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                         RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD |
                         RCC_APB2Periph_GPIOE, DISABLE);


  /* Initialize Leds mounted on MCBSTM32 board */
  // Abilito Clock su GPIO PortB
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

// La linea qui sotto è stata commentata perchè già presente all'inizio del programma
//  GPIO_InitTypeDef GPIO_InitStructure;

  /* Configure the GPIO_LED pin */
  GPIO_InitStructure.GPIO_Pin = (LED8 | LED9 | LED10 | LED11 | LED12 | LED13| LED14|
LED15);
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(LEDPort, &GPIO_InitStructure);

          // LCD Iinit
          lcd_init();
          lcd_clear();
          lcd_print("  Prova_0       ");
          // Posizione, Riga
          set_cursor(0, 1);
          lcd_print("MCBSTM32 lib310 ");
```

Sostituiamo la sezione **while(1)** con quanto scritto sotto:

```
 while (1)
 {
        GPIO_ResetBits(GPIOB, GPIO_Pin_8);
        GPIO_ResetBits(GPIOB, GPIO_Pin_10);
        GPIO_ResetBits(GPIOB, GPIO_Pin_12);
        GPIO_ResetBits(GPIOB, GPIO_Pin_14);
        GPIO_SetBits(GPIOB, GPIO_Pin_9);
        GPIO_SetBits(GPIOB, GPIO_Pin_11);
        GPIO_SetBits(GPIOB, GPIO_Pin_13);
        GPIO_SetBits(GPIOB, GPIO_Pin_15);
  /* Insert delay */
  Delay(0xAFFFF);
        Delay(0xAFFFF);
        Delay(0xAFFFF);

        GPIO_SetBits(GPIOB, GPIO_Pin_8);
        GPIO_SetBits(GPIOB, GPIO_Pin_10);
        GPIO_SetBits(GPIOB, GPIO_Pin_12);
        GPIO_SetBits(GPIOB, GPIO_Pin_14);
        GPIO_ResetBits(GPIOB, GPIO_Pin_9);
        GPIO_ResetBits(GPIOB, GPIO_Pin_11);
        GPIO_ResetBits(GPIOB, GPIO_Pin_13);
        GPIO_ResetBits(GPIOB, GPIO_Pin_15);
  /* Insert delay */
  Delay(0xAFFFF);
        Delay(0xAFFFF);
        Delay(0xAFFFF);

 }
```

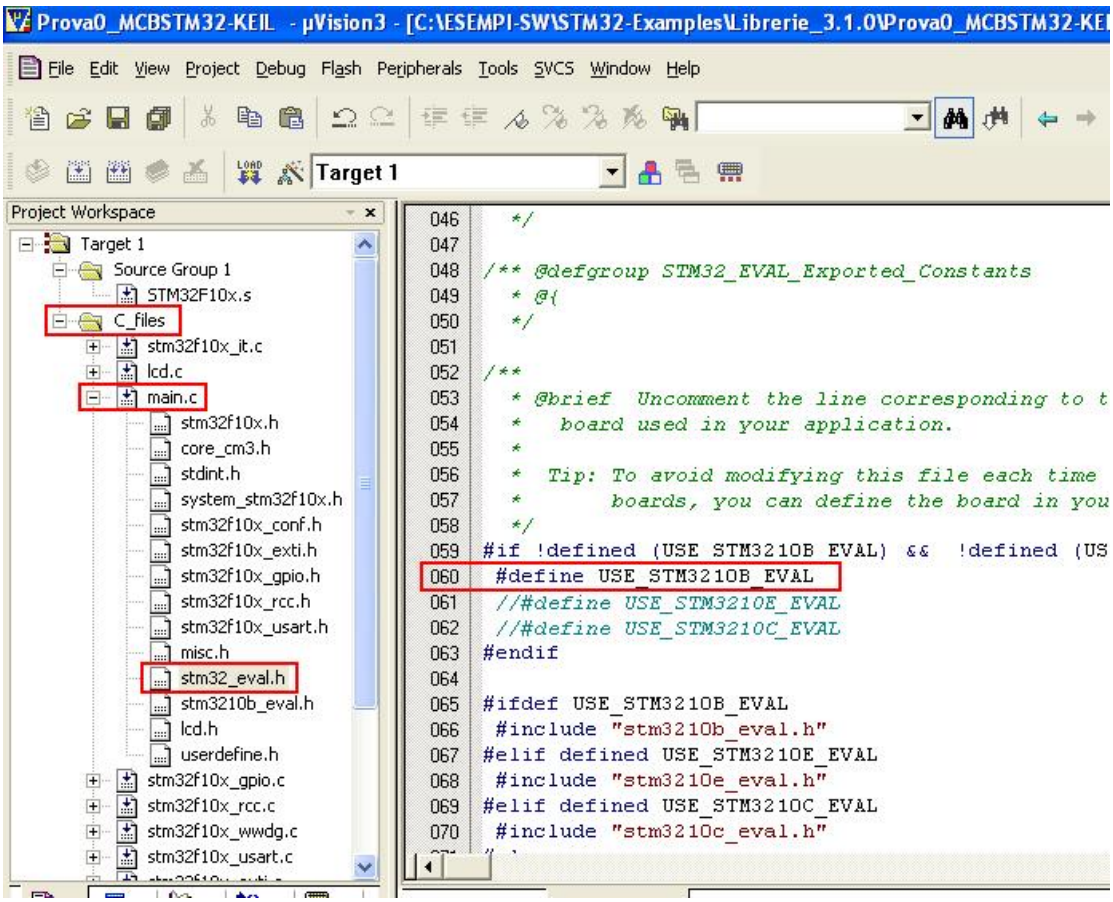
Nella sezione **Includes** ci devono essere gli include qui sotto riportati:

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f10x.h"
#include "stm32_eval.h"
#include "lcd.h"
#include "UserDefine.h"
```

Fatte queste modifiche, prima di proseguire, compiliamo il programma cliccando sull'icona qui sotto riportata.
Questa compilazione ci darà degli errori ma ci serve per avere tutte le dipendenze dei file su cui poi andremo ad agire.

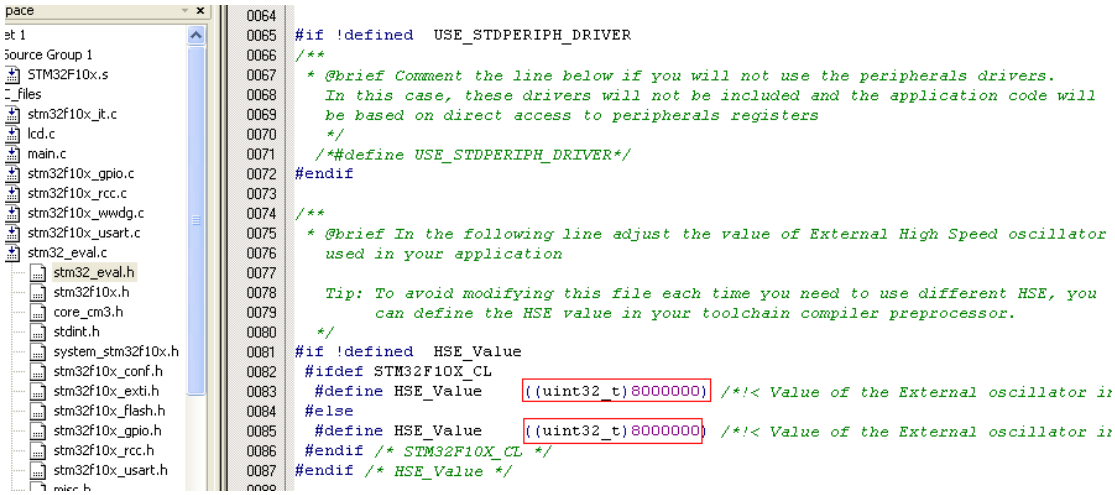
Modifiche a **stm32_eval.h**

Andiamo alla **linea 60** del file e **togliamo il commento alla linea** così da ottenere quanto riportato sotto.



Modifiche a **stm32f10x_conf.h**

Andiamo alle linee 83 e 85 e modifichiamo il valore del quarzo portandolo a 8MHz come sotto evidenziato.



Finalmente abbiamo finito e a questo punto compiliamo nuovamente il programma che ci deve dare 0 errori e 0 warning e poi clicchiamo sull'icona sotto riportata per caricare il programma nella mcu e mandarlo in esecuzione.

Se abbiamo fatto tutto correttamente dobbiamo vedere il **LED lampeggiare** e sul LCD deve comparire la scritta **Prova_0** sulla prima riga mente sulla seconda riga dovrà comparire la scritta **MCBSTM32 lib310**.

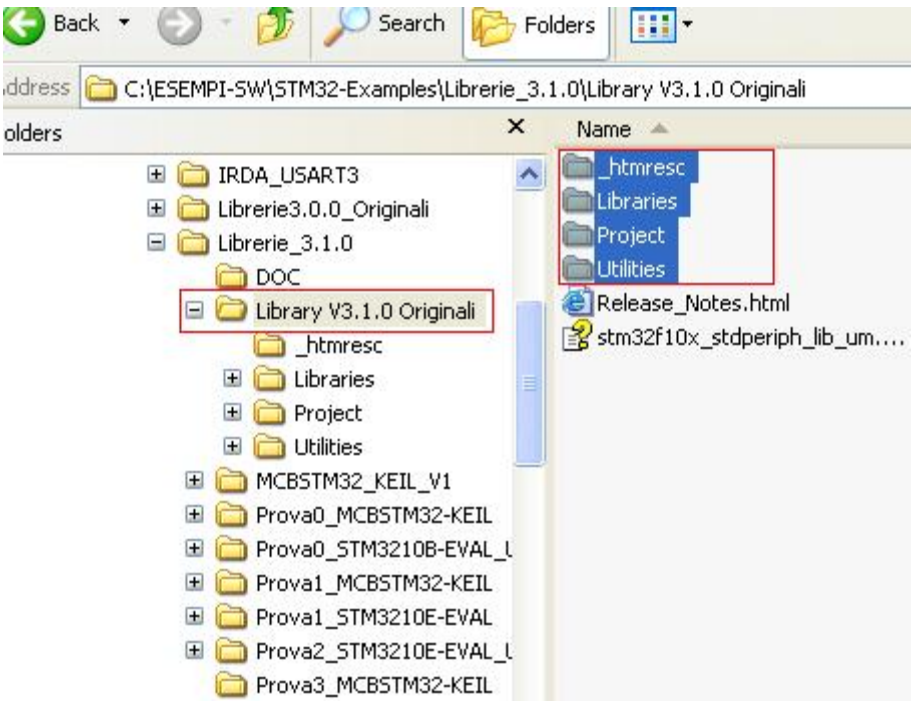
ESEMPIO basato su MCBSTM32 di KEIL partendo usando un esempio delle librerie si STM

(modo semplice d'implementazione)

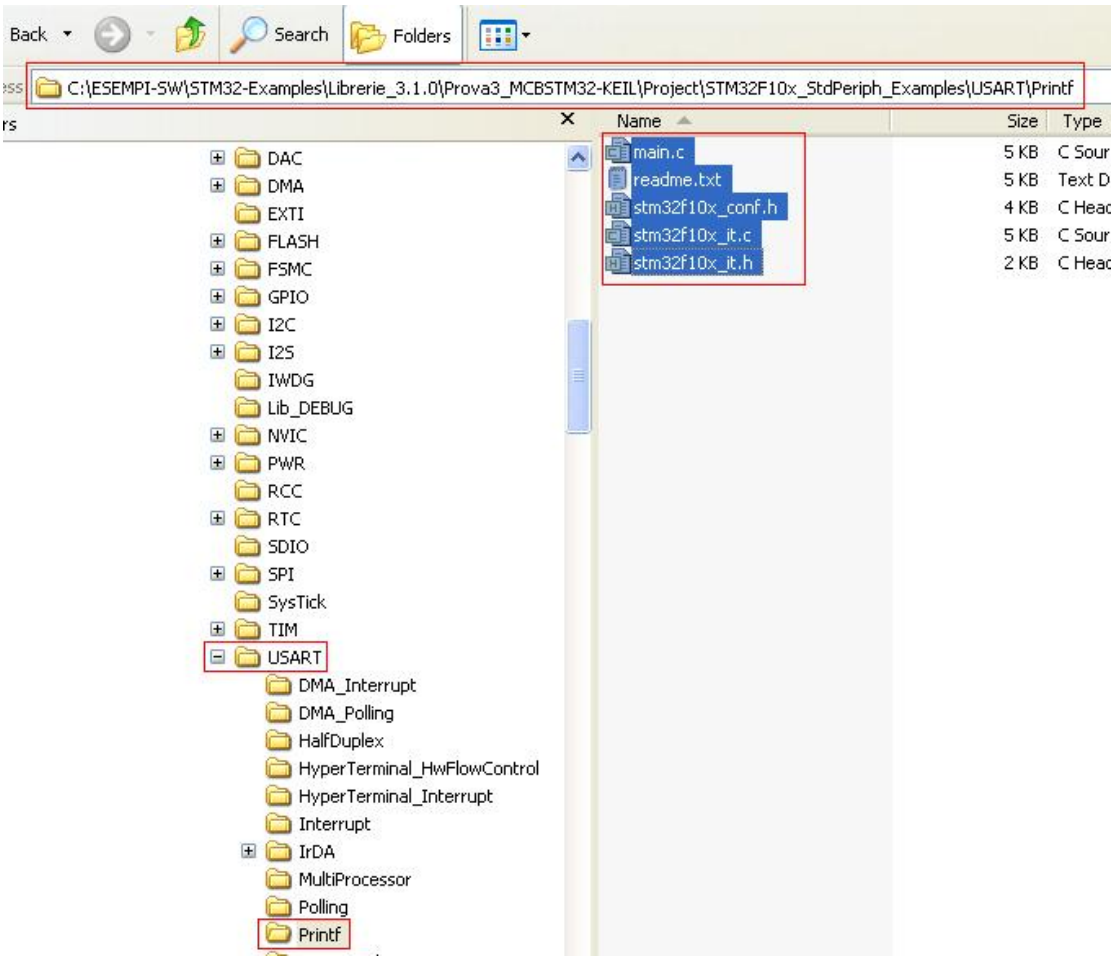
Visto che la MCBSTM32 è in pratica simile alla STM3210B-EVAL di STM per usare gli esempi di STM si può seguire la strada qui sotto dove in pratica si tratta di rimappare solo alcune porte, per esempio i LED, che differiscono tra le due evaluation-board.
Nel nostro caso non servirà rimappare nulla perché useremo la USART1 che su entrambe le evaluation-board è mappata allo stesso modo.

Creiamo una directory che chiameremo **Prova3_MCBSTM32-KEIL** e che nel mio caso si trova qui sotto:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\Prova3_MCBSTM32-KEIL

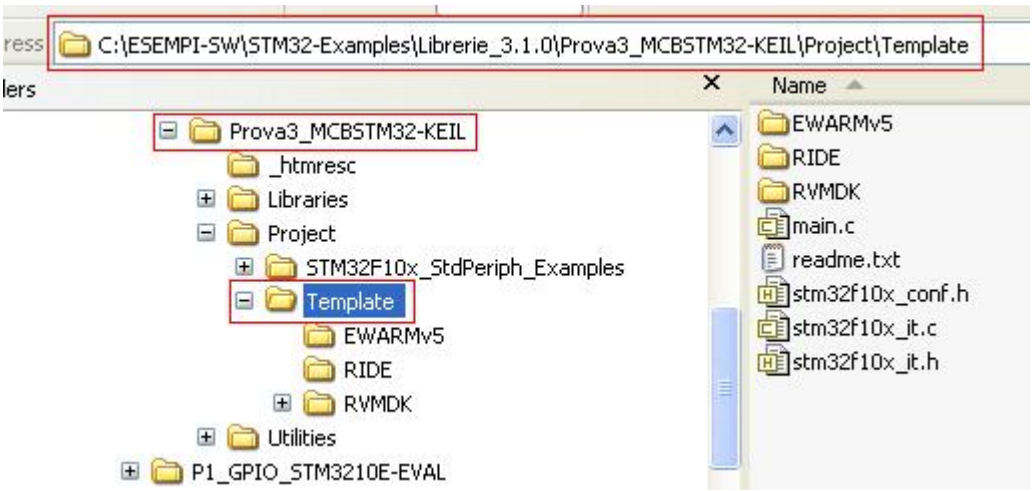
Dalla directory in cui abbiamo scompattato le **librerie 3.1.0** copiamo le cartelle sotto evidenziate nella nostra nuova directory **Prova3_MCBSTM32-KEIL**



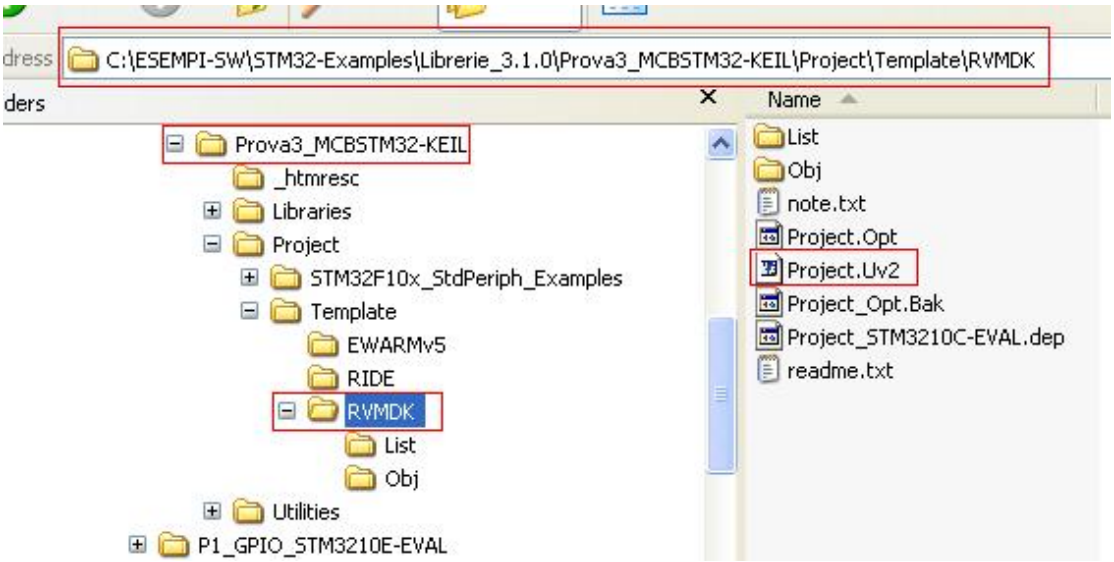
Supponiamo di voler usare l'esempio della **USART** in cui si usa la **printf**.
Per fare ciò dobbiamo copiare i file dell'esempio **…\USART\Printf** che andiamo a prendere dalla directory:
C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0\Prova3_MCBSTM32-KEIL\Project\STM32F10x_StdPeriph_Examples\USART\Printf
(Vedere sotto i rettangoli **rossi**)



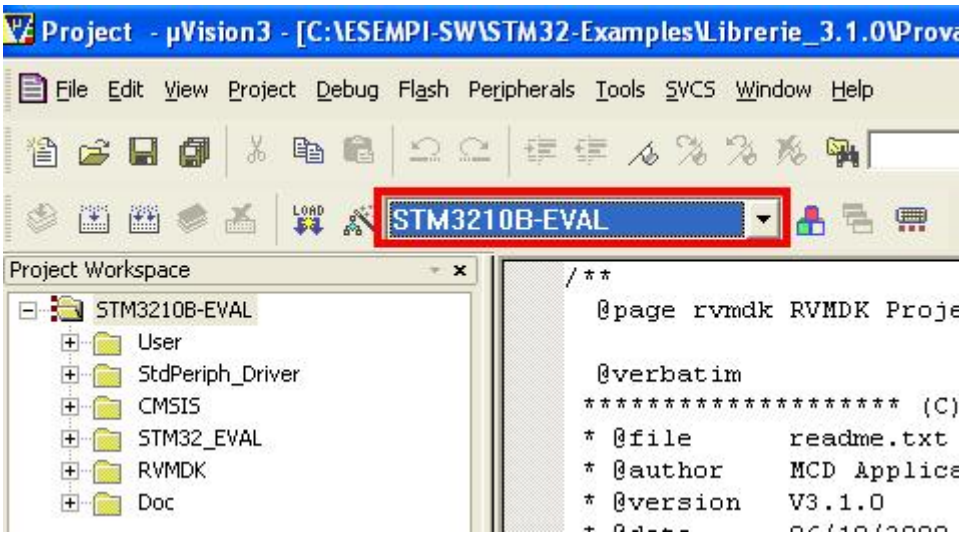
I files da copiare sono:
main.c
readme.txt
stm3210x_config.h
stm3210x_it.c
stm3210x_it.h
vanno copiati nella directory **Template** che si trova sotto la nostra directory di lavoro, in pratica si sovrascrivono i files già presenti.
Vedere sotto.



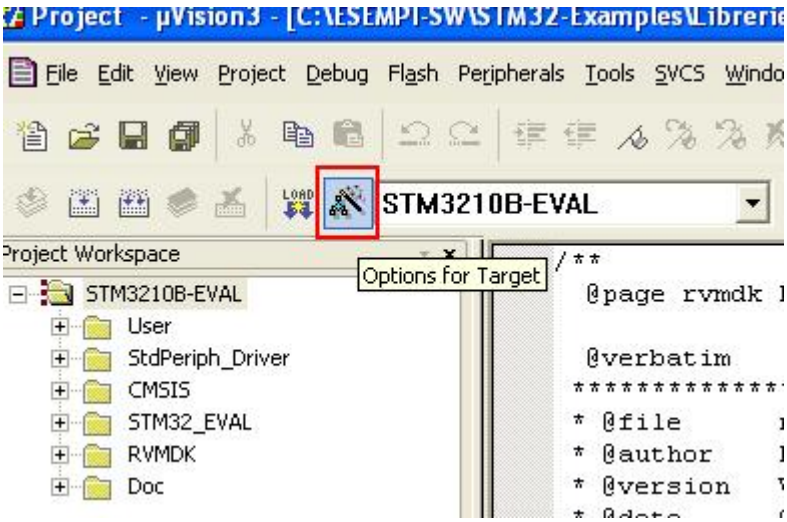
Adesso possiamo aprire il progetto, cliccandoci due volte sopra, che si trova sotto la directory RVMDK e che si chiama **Project.Uv2** come evidenziato sotto.



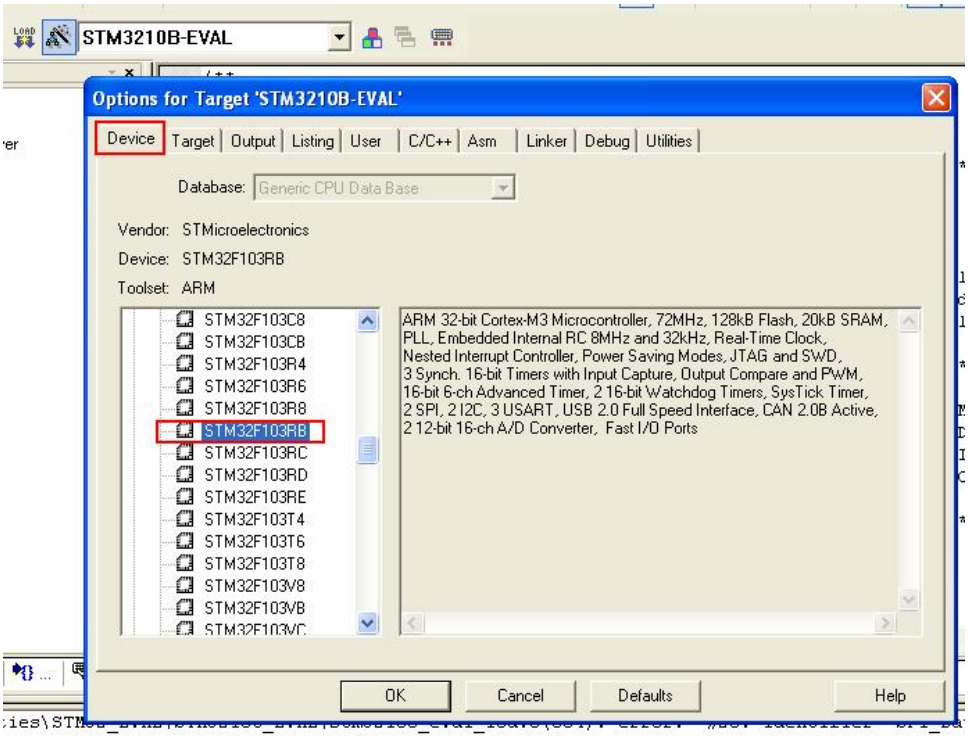
Dalla pagina che compare selezioniamo **STM3210B-EVAL** perché è l'evaluation-board di STM che è similare alla MCBSTM32, vedere sotto.



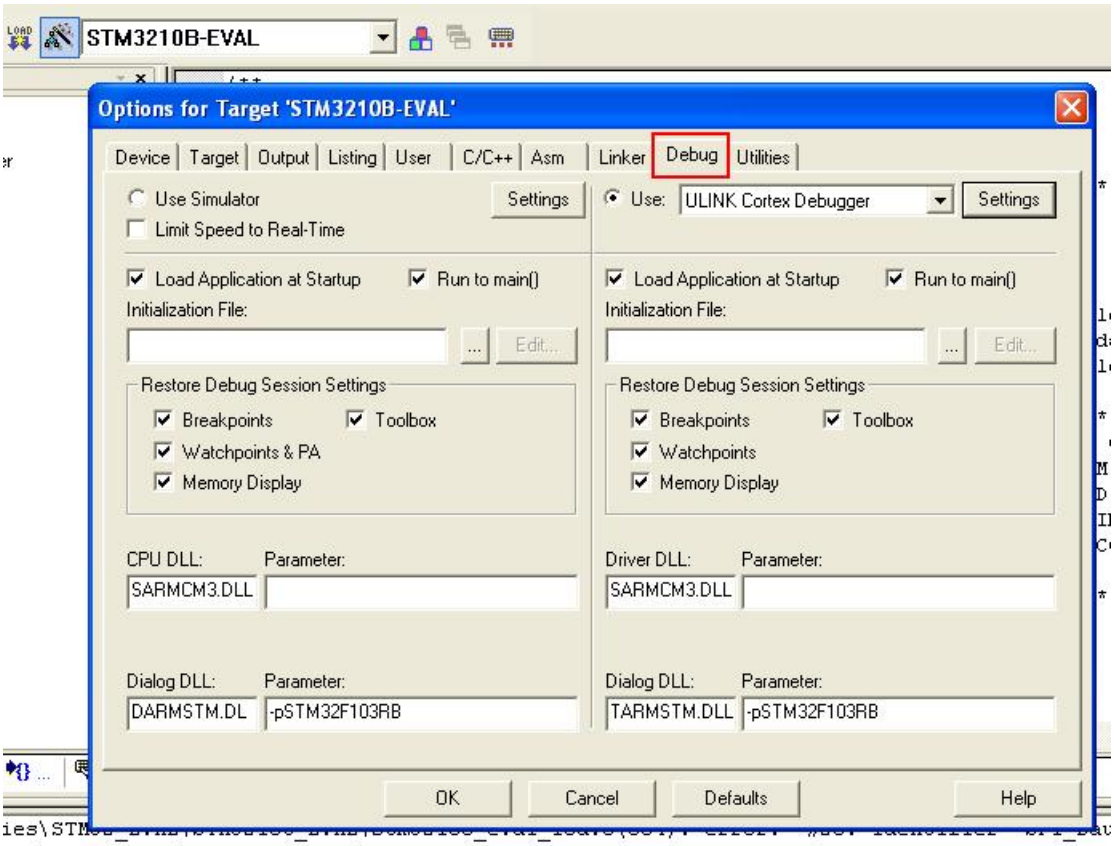
Adesso premiamo sull'icona **Options for Target** come sotto evidenziato.



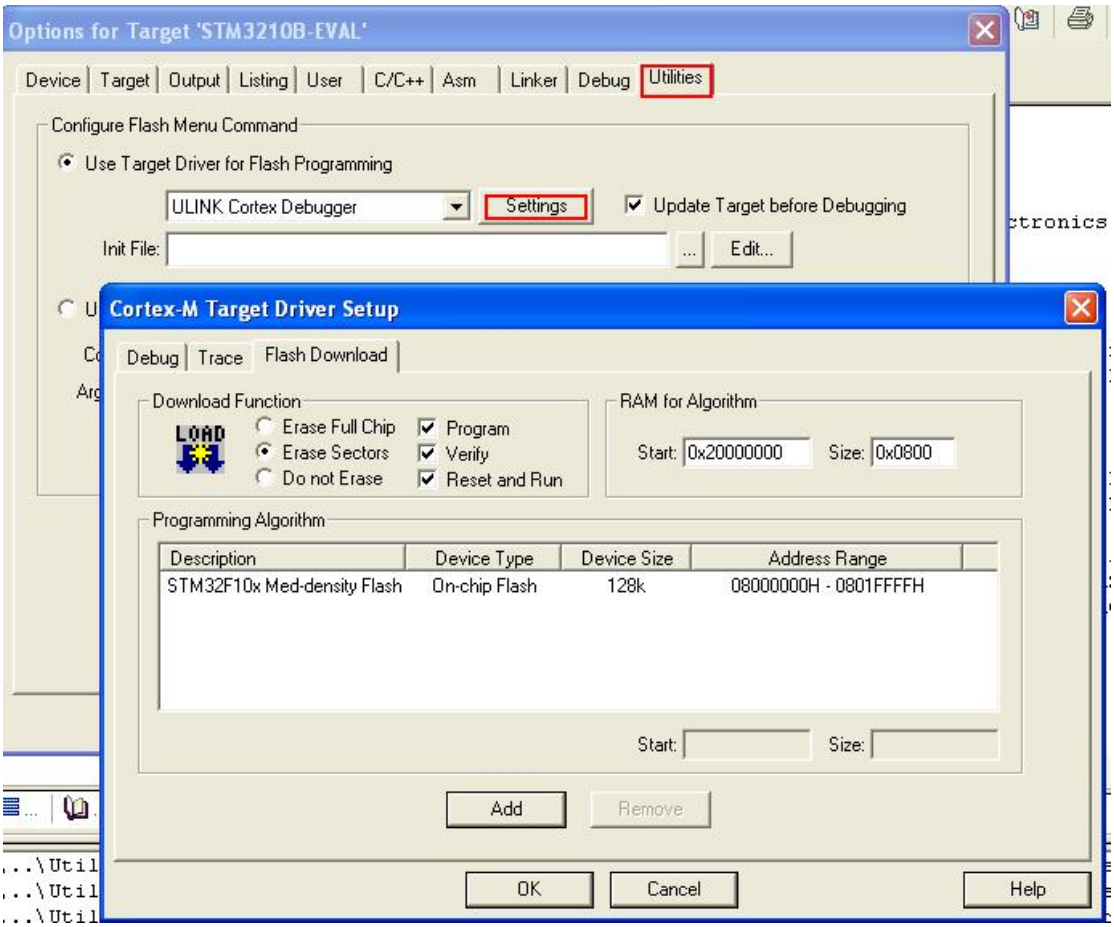
Dalla pagina che compare premiamo su **Device** e selezioniamo **STM32F103RBT6** come sotto evidenziato.



Ora premiamo su **Debug** e assicuriamoci che la pagina sia configurata come sotto.



Come ultimo passaggio premiamo **Utilities** e poi su **Settings** e assicuriamoci che le pagine siano configurate come sotto.



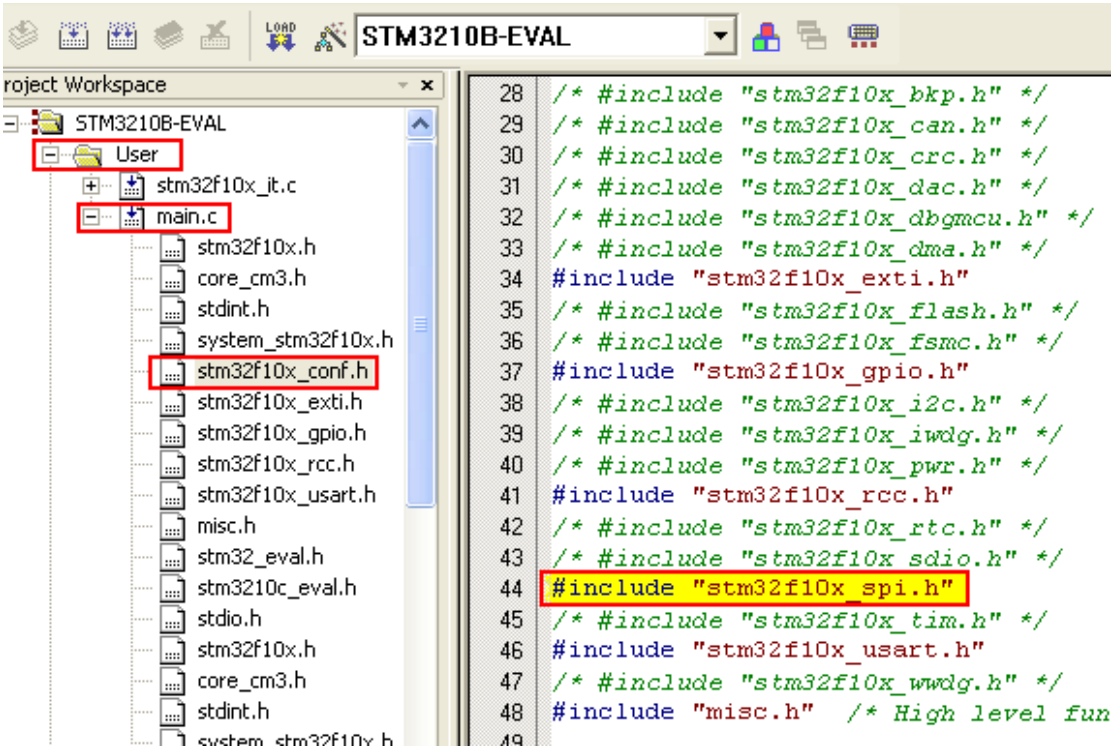
Per terminare diamo due volte OK.

Se adesso **compiliamo**, premere l'icona a lato  e dobbiamo avere degli errori perché serve ancora fare delle inclusioni qui sotto descritte.
Abbiamo compilato per avere tutte le dipendenze dei file che adesso andremo a modificare

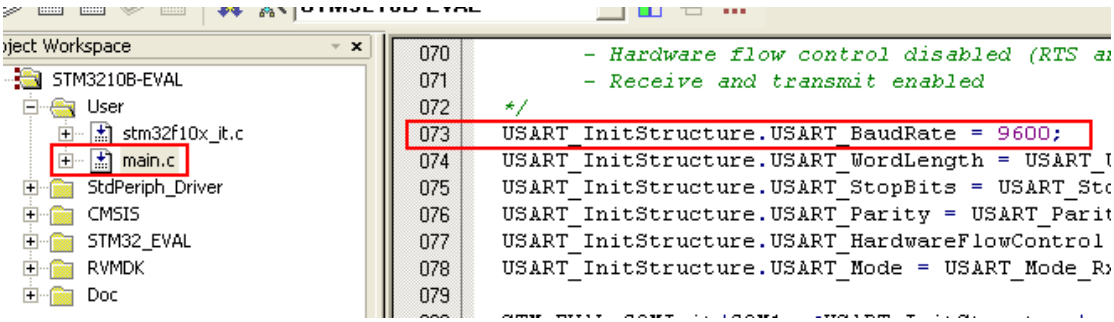
Sotto la cartella **User** c'è **main.c** e sotto le sue dipendenze c'è il file **stm32f10x_conf.h** che va aperto per controllare che la riga:

44 #include "stm32f10x_spi.h"

rettangolo rosso a sfondo giallo, NON sia sotto commento.



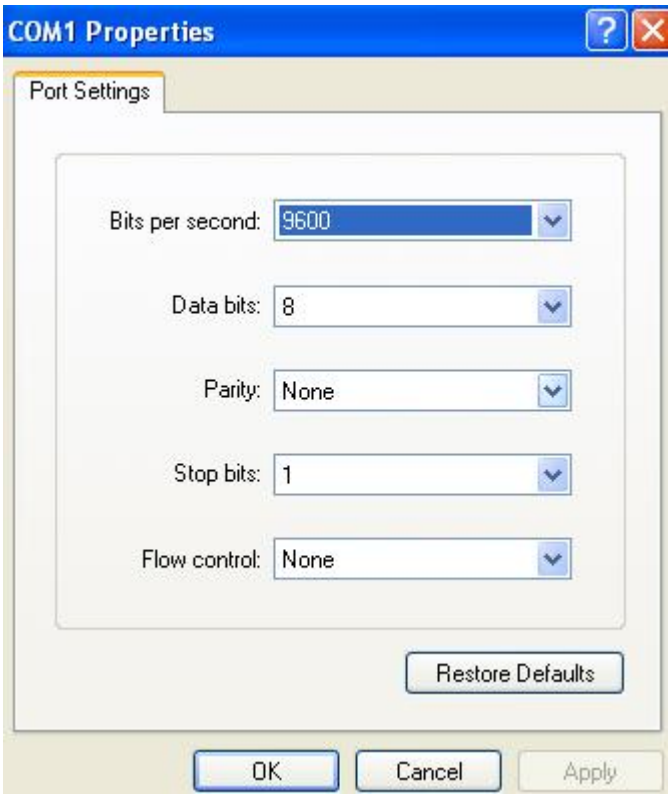
Come ultima modifica andiamo ad aprire **main.c** e cambiamo la velocità di comunicazione dell'USART e portiamola a **9600** come sotto evidenziato.



Abbiamo finito e adesso possiamo compilare premendo l'icona:

Se abbiamo fatto tutto correttamente avremo 0 errori e 0 warning.

Per verificare la funzionalità del programma apriamo **HyperTermianl** e configuriamolo a 9600,8,N,1,N (vedere sotto).



Colleghiamo un cavo “dritto” tra la porta marchiata **COM** della **MCBSTM32** e la seriale del PC.
I collegamenti sono i seguenti:

Pin.2 con Pin.2
Pin.3 con Pin.3
Pin.5 con Pin.5

Per mandare in esecuzione il programma premete sull’icona sotto

Al termine del caricamento del programma su HyperTerminal vi deve comparire la scritta sotto:

```
USART Printf Example: retarget the C library printf function to the USART
```

Se avete voglia potreste includere nel vostro progetto le librerie di KEIL per la gestione del LCD (presente sulla scheda MCBSTM32) e che sono date in dotazione da KEIL per fare comparire delle scritte sull'LCD.
Nel file ZIP allegato questa implementazione è stata fatta.