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 pg.1

ESEMPIO basato su MCBSTM32 di KEIL partendo creando un nuovo progetto da zero pg.**8**

ESEMPIO basato su MCBSTM32 di KEIL partendo usando un esempio delle librerie si STM *(modo semplice d'implementazione)* pg.**29**

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.

Prova1 Prope	rties 🔹 🕐 🔀
General Shari	ng Security Customize
	Prova1
Туре:	File Folder
Location:	C:\ESEMPI-SW\STM32-Examples\Librerie_3.1.0
Size:	16,9 MB (17.787.348 bytes)
Size on disk:	18,1 MB (19.066.880 bytes)
Contains:	588 Files, 135 Folders
Created:	domenica 19 luglio 2009, 1.16.01
Attributes:	Read-only Advanced
	OK Cancel Apply

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





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.

The second se		
Eile Edit View Project Debug Flash Peri	pherals <u>T</u> ools <u>S</u> VCS <u>W</u> ir	ndow <u>H</u> elp
12 😅 🖬 🌒 🕹 🖻 🖻 🗠 🕰	∰ ∰ ∧ % %	% 🙀
🗇 🖄 🛅 🥌 👗 🙀 🌋 STM3210	DE-EVAL	- 🔒 🚍 📟
Project Workspace	/ * *	
E STM3210E EVAL	<pre>@page rvmd)</pre>	t RVMDK Project T
🗄 📄 User 🔪		
🗄 🚞 StdPeriph_Driver	0verbatim	
E CMSIS	*********	********* (C) COP
E STM32_EVAL	* 0file	readme.txt
🕀 🛅 RVMDK	* @author	MCD Application
🗄 📄 Doc	* @version	V3.1.0
	* @date	06/19/2009
	* @brief	This sub-direct
	*	to create a new
	*	Library and wor
	*	software toolch
	********	* * * * * * * * * * * * * * * * * *
	* THE PRESEN	NT FIRMWARE WHICH
	* WITH CODIN	IG INFORMATION RE

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.

🗗 Project - μVision3 - [C:\ESEMPI-SW\S	TM32-Examples\Librerie_3.1.0\Prova1\Project\Template\stm32f10x_conf.h]
Eile Edit View Project Debug Flash Pe	ripherals <u>T</u> ools <u>S</u> VCS <u>Wi</u> ndow <u>H</u> elp
12 C 3 d × 1 🕼 🖬 📽 2 C	拝 拝 ⊿ % % % % → (2 ●
🗇 🕮 📾 🚿 🕌 🙀 🔊 STM321	OE-EVAL 🔄 📇 🖷
Project Workspace 🔹 👻	25 /* Includes
STM32_EVAL	26 /* Uncomment the line below to enable peripheral header f
🖃 🔝 stm32_eval.c	27 /* #include "stm32f10x_adc.h" */
stm32_eval.h	28 /* #include "stm32f10x_bkp.h" */
🔜 stm32f10x.h	29 /* #include "stm32f10x_can.h" */
- 🔜 core_cm3.h	30 /* #include "stm32f10x_crc.h" */
🔜 stdint.h	31 /* #include "stm32f10x_dac.h" */
system_stm32f10x.h	32 /* #include "stm32f10x_dbgmcu.h" */
stm32f10x_conf.h	33 /* #include "stm32f10x_dma.h" */
stm32f10x_exti.h	34 #include "stm32f10x exti.h"
stm32f10×_gpio.h	35 /* #include "stm32f10x flash.h" */
stm32f10x_rcc.h	36 #include "stm32f10x_fsmc.h"
stm32f10×_usart.h	37 #include "stm32f10x gpio.h"
misc.h	38 /* #include "stm32f10x_i2c.h" */
stm3210c_eval.h	39 /* #include "stm32f10x_iwdg.h" */
stm32_eval.h	40 /* #include "stm32f10x_pwr.h" */
🔜 stm32f10×.h	41 #include "stm32f10x rcc.h"
core_cm3.h	42 /* #include "stm32f10x_rtc.h" */
stdint.h	43 /* #include "stm32f10x_sdio.h" */
system_stm32f10x.h	44 #include "stm32f10x_spi.h"
stm32f10x_conf.h	45 /* #include "stm32f10x_tim.h" */
stm32f10x_exti.h	46 #include "stm32f10x usart.h"
- 🔜 stm32f10x_fsmc.h	47 /* #include "stm32f10x_wwdg.h" */
stm32f10×_gpio.h	48 #include "misc.h" /* High level functions for NVIC and S
🔜 stm32f10x_rcc.h	49
🚽 🔜 stm32f10x_spi.h 🥫	

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.

🌃 Project - µVision3 - [C:\ESEMPI-SW\S	TM32-Examples\Librerie_3.1.0\Prova1
File Edit View Project Debug Flash Per	ipherals <u>T</u> ools <u>S</u> VCS <u>W</u> indow <u>H</u> elp
1 🗃 🚅 🖬 🕼 👗 🛍 🛍 🗅 🕰	₩ # # & % % % % %
🔮 🕮 🕮 🥌 🔏 💥 💦 STM321	0E-EVAL 💽 🛃 🖷 📖
Project Workspace	/* Includes
STM32_EVAL	Zb /* Uncomment the line be.
😑 🔝 stm32_eval.c	27 /* #include "stm32f10x_a
stm32_eval.h	28 /* #include "stm32f10x_b;
	29 /* #include "stm32f10x_c.
core_cm3.h	30 /* #include "stm32f10x_c.
stdint.h	31 /* #include "stm32f10x_d.
system_stm32f10x.h	32 /* #include "stm32f10x_di

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.

Elle Edit View Project Debug Flash Peripherals Iools SVCS Window Help	
Options for Target 'STM3210E-EVAL'	(1) - (2) -
S Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Proj. Configure Flash Menu Command	
• Use Target Driver for Flash Programming	ral header file inclusi
ULINK Cortex Debugger Settings Vpdate Target before Debugging	
Init File: Cortex-M Target Driver Setup	
C Use External Tool for Flas Debug Trace Flash Download	
Command: RAM for Algorithm	
Arguments: C Erase Full Chip V Program Start: 0x20000000 S	Size: 0x0800
Run Indep	
Programming Algorithm	
Description Device Type Device Size Address	Range
STM32F10x High-density Flash On-chip Flash 512k 08000000H -	0807FFFFH
Start: Sta	Size:
Add Remove	
E Cancel	Help

× ".\Obj\STM3210E-EVAL.axf" - 0 Error(s), 0 Warning(s).

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

🌃 Project - µVision3 - [C:\ESEMPI-SW\ST	M32	Examples\Libro
Eile Edit View Project Debug Flash Perip	herals	<u>T</u> ools <u>S</u> VCS <u>W</u> i
12 C 18 8 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	匪	i≡ ∧ % %
🗇 🕮 🐲 📥 🧱 🔊 STM3210	E-EV	AL
Project Workspace Nownload to Flash Me	mory	/* Includes
stm32f10x_conf.h	26	/* Uncommer
stm32f10x_exti.h	27	/* #include
stm32f10x_fsmc.h	28	/* #include
stm32f10x_gpio.h	29	/* #include
	30	/* #include
stm32f10x_spi.h	31	/* #include
stm32f10x_usart.h	32	/* #include
misc.h	33	/* #include
stm3210e_eval.h	34	#include "s
🕂 🛄 stm3210c_eval_lcd.c	35	/* #include
🗄 🔝 stm3210e_eval_lcd.c	36	#include "s
stm3210e eval lcd.h	37	#include "s

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: **Icd.c** e **Icd.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.

🕎 µVision3	
<u>File E</u> dit <u>V</u> iew Pro	pject Debug Flash Peripherals Tools SVCS Window Help
🖀 🛩 🖬 🕻	New µYision Project
	New Project <u>W</u> orkspace
Project Workspace	Import µVision1 Project
Froject workspace	Open Project
	Close Project
	Manage
	Select Device For Target (STM2210E_EVAL)

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:

Icd.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.

- ---- 📩 stm32f10x_gpio.c
- stm32f10x_wwdg.c
- --- 🔝 stm32f10x_usart.c --- 🔝 stm32f10x_exti.c
- ---- 🔝 stm32h10x_exti.
- stm32f10x_spi.c
- 🔝 stm32f10×_fsmc.c

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
                 GPIOB
#define LEDPort
#define LED8
                 GPIO_Pin_8
#define LED9
                 GPIO Pin 9
#define LED10
                 GPIO Pin 10
#define LED11
                 GPIO Pin 11
                 GPIO Pin 12
#define LED12
#define LED13
                 GPIO Pin 13
                 GPIO Pin 14
#define LED14
                 GPIO Pin 15
#define LED15
```

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.

🕎 Prova0_MCBSTM32-KEIL - μVision3	- [
Eile Edit View Project Debug Flash Pe	eripl
🎦 😂 🖬 🎒 👗 🖻 🛍 🗅 🕰	[
	R
😂 🕮 🔎 🔏 🙀 💦 Target	1
Project Workspace 🔹 👻	
Image: 1 Image: Source Group 1 Image: STM32F10x.s Image: Stm32f10x_it.c Image: Stm32f10x_it.c Image: Stm32f10x_gpio.c Image: Stm32f10x_gpio.c Image: Stm32f10x_gpio.c Image: Stm32f10x_wwdg.c Image: Stm32f10x_wwdg.c Image: Stm32f10x_spi.c Image: Stm32f10x_spi.c <td></td>	

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.



Prova0_MCBSTM32-KEIL - µV	fision3	
Project Workspace	Shi regulierals gools	
→ Target 1 → Target 1 → STM32F10x.s → Cfiles → Icd.c → stm32f10x_gpio.c → stm32f10x_gpio.c → stm32f10x_wwdg.c → stm32f10x_sext.c → stm32f10x_sext.c	Device Target Dutput Listing User C/C++ Asm Linker Debug Utilities Preprocessor Symbols Define: Undefine: Language / Code Generation Strict ANSI C Warnings: Optimization: Level 0 (-00) Enum Container always int Quescified> Plain Char is Signed Thomb Mode Optimize for Time Plain Char is Signed Thomb Mode Split Load and Store Multiple Read-Only Position Independent Thomb Mode Include Read-Write Position Independent Include Paths Misc Controls Controls Compiler co-device DARMSTM -g-00apcs=interwork -1 "C:\Keil\ARM\INC\ST\STM32F10x" -o "".o" Image: String Image: String	
	OK Cancel Defaults Help	

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_Driv er\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.

🕎 Prova0_MCBSTM32-KEIL - μ\	/ision3
File Edit View Project Debug Fla	sh Perjpherals Iools SVCS Window Help 으으 译字 4 % % % % % [] [] [] [] [] [] [] [
Project Workspace	
Target 1 Source Group 1 Source Group 1 Stm32F10x.s Stm32F10x.jt.c Istm32f10x_groio.c stm32f10x_groio.c stm32f10x_groio.c stm32f10x_wwwdg.c stm32f10x_wwdg.c stm32f10x_wsdi.c stm32f10x_gsdi.c	Options for Target 'Target 1' Device Target Output Listing User C/C++ Asm Linker Debug Utilities Preprocess Setup Compiler Include Paths: Define: Proval_MCBSTM32:KEIL Undefine: Proval_MCBSTM32:KEIL Undefine: Libraries\CMSIS\Core\CM3 Libraries\STM32:EVAL Ubitraries\STM32:EVAL Optimization Optimization OK Controls
	OK Cancel Defaults Help

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.

🕎 ProvaO_MCBSTM32-KEIL - µV	ision3
Eile Edit View Project Debug Flas	h Peripherals <u>T</u> ools <u>S</u> VCS <u>Wi</u> ndow <u>H</u> elp
12 6 9 9 1 8 6	
🛛 🕸 🕮 🥌 🔏 💥 🛣 T	Farget 1 🔄 🛃 🖷
Project Workspace	
🖃 🔁 Target 1	Options for Target 'Target 1'
E Source Group 1	Device Target Output Listing User C/C++ Asm Linker Debug Utilities
stm32f10x it.c	Preprocessor Symbols
📰 lcd.c	Definer USE STDPERIPH DRIVER, STM32F10X MD, USE STM3210B EVAL
main.c	
stm32f10x_gpio.c	Undeline
stind2110x_vecte	Language / Code Generation
stm32f10x_usart.c	🗖 Strict ANSI C 🛛 🛛 🖓 arnings:
stm32f10x_exti.c	Optimization: Level 0 (-00) 👻 🔽 Enum Container always int 🦳 🗸 🗸
stm32f10x_spi.c	Optimize for Time Plain Char is Signed
stm32_eval.c	Split Load and Store Multiple Read Only Position Independent
- 🔛 core_cm3.c	Dre ELE Section per Function E Read-Write Position Independent
misc.c	
userberine.n	Include\Prova0_MCBSTM32:KEIL;\Libraries\CMSIS\Core\CM3;\Libraries\STM32F10x_StdPeriph_Driver
	Mise
	Controls
	Compiler -cdevice DARMSTM -g -O0apcs=interwork -l\Prova0_MCBSTM32-KEIL
	control -I.\Libraries\CMSIS\Core\CM3 -I.\Libraries\STM32F10x_StdPeriph_Driver\inc
	OK Cancel Defaults Help

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

🕎 ProvaO_MCBSTM32-KEIL — j	iVision3
Eile Edit View Project Debug Fl	ash Peripherals Tools SVCS Window Help
12 😅 🖬 🕼 👗 🖻	
🖉 🎞 🗰 🖉 者 🛛 🎬 🌋	Target 1 🗾 🛃 🚍 📰
Project Workspace	Options for Target 'Target 1'
Image: 1 Image: Source Group 1 Image: STM32F10x.s Image: STM32F10x_jt.c Image: Stm32f10x_gpio.c Image: S	Device Target Output Listing User C/C++ Asm Linker Debug Utilities Configure Flash Menu Command © Use Target Driver for Flash Programming ULINK Cortex Debugger Settings V Update Target before Debugging Init File: Edit © Use External Tool for Flash Programming Command: Arguments: Run Independent.
 .	OK Cancel Defaults Help

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

🔽 Prova0_MCBSTM32-KEIL - µVision3							
Eile Edit View Project Debug Flash Peripherals Tools SVCS Window Help							
曾 🖨 🖬 🕼 🕹 의 으 🛛 🕸 博 🦽 🌾 🌾 🧏 🦓 🦓 🦓 🔛 🔙 💆 🗮 🖉 🖉							
🤌 🕮 🖉 🝝 🙀 💦 Target 1 💽 🛃 🐂							
Project Workspace Options for Target 'Target 1'							
Target 1 Target 1 Source Group 1 STM32F10x.s C files	Cortex-M Target Driver Setup						
 Image Straight St	Download Function C Erase Full Chip ✓ Program C Erase Sectors ✓ Verify ✓ Start: 0x20000000 C D on ot Erase ✓ Reset and Run						
	Programming Algorithm Description Description Description STM32F10x Med-density Flash On-chip Flash 128k 08000000H - 0801FFFFH						
UserDefine.h	Start: Size:						
	Add Remove						
	OK Cancel Help						
B B 1	OK Cancel Defaults Help						

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)

RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOB | 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 linit 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.

pace ~	×	0064	
ət 1	^	0065	#if !defined USE_STDPERIPH_DRIVER
Source Group 1		0066	/**
STM32F10x.s		0067	* @brief Comment the line below if you will not use the peripherals drivers.
I_files		0068	In this case, these drivers will not be included and the application code will
stm32f10×_it.c		0069	be based on direct access to peripherals registers
🟦 lcd.c		0070	*/
📩 main.c		0071	/*#define USE_STDPERIPH_DRIVER*/
stm32f10×_gpio.c		0072	#endif
stm32f10×_rcc.c		0073	
🟦 stm32f10×_wwdg.c	=	0074	/**
stm32f10×_usart.c		0075	* Gbrief In the following line adjust the value of External High Speed oscillator
stm32_eval.c		0076	used in your application
stm32_eval.h		0077	
🔜 stm32f10×.h		0078	Tip: To avoid modifying this file each time you need to use different HSE, you
core_cm3.h		0079	can define the HSE value in your toolchain compiler preprocessor.
		0080	*/
system_stm32f10x.h		0081	#if !defined HSE_Value
stm32f10×_conf.h		0082	#ifdef STM32F10X_CL
stm32f10x_exti.h	-	0083	#define HSE_Value ((uint32_t)8000000) /*!< Value of the External oscillator in
stm32f10×_flash.h		0084	#else
stm32f10×_gpio.h		0085	#define HSE_Value ((uint32_t)8000000) /*!< Value of the External oscillator in
stm32f10x_rcc.h		0086	#endif /* STM32F10X_CL */
stm32f10×_usart.h		0087	<pre>#endif /* HSE_Value */</pre>
ind 1 mice b		0000	

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.

🕻 Project - µVision3 - [C:\ESEMPI-SW\STM32-Examples\Librerie Eile Edit View Project Debug Flash Peripherals Tools SVCS Windo X B 6 诌 🚅 🔒 🕼 I 🔁 🍽 🧶 X LOAD 2 STM3210B-EVAL • Project Workspace · · Options for Target E STM3210B-EVAL @page rvmdk 1 🗄 🦳 User 🗄 🚞 StdPeriph_Driver Rverbatim 🗄 🛅 CMSIS *********** 🗄 🚞 STM32_EVAL Ofile 1 🗄 🚞 RVMDK @author 1 🛨 📄 Doc **Øversion** 3 * 0 data

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

	TIONS IOF TAIget STM52TOD-LYAL	
L	Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
	Database: Generic CPU Data Base	
	Vendor: 51 Microelectronics	
	Device: STM32FTU3RB	
	AFM 32-FU3L8 AFM 32-bit Lottex-M3 Microcontroller, 72MHz, 128kB Flash, 20kB SRAM, PLL, Embedded Internal RC 8MHz and 32kHz, Real-Time Clock,	^
	STM32F103R4 Nested Interrupt Controller, Power Saving Modes, JTAG and SWD, 3 Supply 16-bit Timers with Input Capture, Dutput Compare and PW/M	
	STM32F103R6 16-bit 6-ch Advanced Timer, 216-bit Watchdog Timers, SysTick Timer,	
	2 SPI, 212C, 3 USART, USB 2.0 Full Speed Interrace, LAN 2.08 Active, 212-bit 16-ch A/D Converter, Fast I/O Ports	
	STM32F103RC	
	G STM32F103RD	
	CI STM32F103HE	
	G STM32F103T8	
	CI STM32F103V8	0
	CI STM32F103VC 🕑 🔄	

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

Linker Debug Hitilities							
Device Target Output Listing User C/C++ Asm Linker Debug Utilities							
s 🔍 Use: ULINK Cortex Debugger 💽 Setting							
✓ Load Application at Startup ✓ Run to main() Initialization File:							
Edit							
Restore Debug Session Settings							
🔽 Breakpoints 🔽 Toolbox							
₩ Watchpoints							
Memory Display							
Driver DLL: Parameter:							
SARMCM3.DLL							
Dialog DLL: Parameter:							

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

Options for Target 'STM3210B-EVAL'	🛛 🕅 🛃
Device Target Output Listing User C/C++ Asm Linker Debug Utilities Configure Flash Menu Command ULINK Cortex Debugger ULINK Cortex Debugger Init File: Little Litt	rtronics
C U Cortex-M Target Driver Setup	
Co Debug Trace Flash Download Arg Download Function RAM for Algorithm Co Erase Full Chip Image: Program Co Erase Sectors Image: Verify Co Do not Erase Image: Reset and Run Programming Algorithm Programming Algorithm	
Description Device Type Device Size Address Range STM32F10x Med-density Flash On-chip Flash 128k 08000000H - 0801FFFFH	
Start: Size:	
Add Remove	
\Util	Help

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).

M1 Properties	?
ort Settings	
Bits per second:	9600
Data bits:	8
Parity:	None
Stop bits:	1
Flow control:	None
	Restore Defaults
0	K Cancel Apply

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

LOAD

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.