

UNIVERSITA’ DEGLI STUDI DI SALERNO

**Facoltà di ingegneria**

Corso di laurea in ingegneria elettronica

PROGETTO IN

**Misure per L’Automazione**

Manuale Programmatore

**STUDENTE**

Flavio Della Calce

**INSEGNANTE**

Prof. Vincenzo Paciello

**A.A 2021/2022**

1. **Introduzione**

1.1 Descrizione del software realizzato.

1. **Caratteristiche del software**

2.1 Descrizione dei registri utilizzati.

2.2 Descrizione delle funzioni sviluppate.

2.3 Diagramma a blocchi del software realizzato.

1. **INTRODUZIONE**

**Descrizione del software realizzato.**

Il software realizzato ha come obiettivo quello di:

* Inizializzare una comunicazione seriale con il microcontrollore;
* Consentire l’attivazione di segnali PWM;
* Consentire la modifica dei parametri dei segnali PWM attivi;
* Consentire la variazione del baud rate;
* Consentire la lettura dell’ID del microcontrollore;

Per poter implementare queste funzionalità sono stati utilizzate diverse periferiche e sono stati definite diverse funzioni che verranno illustrate nel capitolo successivo. Il programma utilizzato per la realizzazione del software è Keilµvision, un software progettato per tutti i microcontrollori con architettura ARM cortex. In particolare, il microcontrollore su cui è stato implementato il software realizzato è l’STM32 NUCLEO F401RE. Per ulteriori informazioni riguardanti le funzionalità offerte e la scheda di sviluppo utilizzata consultare il manuale utente.

1. **CARATTERISTICHE DEL SOFTWARE**

**2.1 Descrizione dei registri utilizzati.**

*In questa sezione vengono descritti i registri utilizzati e le loro funzionalità.*

***RCC***



L’RCC (Reset and Clock Control) è un registro essenziale la cui funzione è quella abilitare il clock alle differenti periferiche utilizzate. I due registri dichiarati consentono di abilitarlo sulle periferiche da noi utilizzate, ovvero APB1 e AHB1. Si noti che il clock non è abilitato di default a ogni periferica per limitare il consumo della scheda.

***GPIOA***

Per GPIOA si intende “General Purpose Input/Output” relativo alla porta A, ovvero i pin della scheda di sviluppo che costituiscono tale porta.

Nello sviluppo del software sono stati utilizzati tre registri che appartengono alla categoria GPIOA, ovvero:

1. GPIOA\_MODER
2. GPIOA\_ODR
3. GPIOA\_AFRL
4. ***GPIOA\_MODER***



Il registro GPIOA\_MODER definisce la modalità (ingresso/uscita/alternata) relativa al singolo pin, in particolare nella progettazione della scheda di sviluppo sono stati utilizzati sei pin di cui quattro in modalità output per fornire l’onda quadra relativa ai segnali PWM e gli altri due per implementare la comunicazione seriale in modalità alternata (Rx e Tx).

1. ***GPIOA\_ODR***



Il registro GPIOA\_ODR contiene tutti gli stati di output dei pin della porta GPIOA e serve quindi a gestire il bit in uscita e viene utilizzato nella generazione dell’onda quadra relativa ai segnali PWM.

1. ***GPIOA\_AFRL***



Il registro GPIOA\_AFRL serve a selezionare il particolare tipo di funzione alternata sul singolo pin; in questo caso è stato utilizzato il register LOW in quanto stiamo utilizzando in modalità alternata i pin PA2 e PA3 che ricadono nella categoria pin LOW ovvero da PA0 fino a PA7.

***USART2***

L’USART è una periferica del microcontrollore adibita alla gestione della comunicazione seriale.

Nello sviluppo del software sono stati utilizzati quattro registri che appartengono alla categoria USART2, ovvero:

1. USART2\_CR1
2. USART2\_BRR
3. USART2\_SR
4. USART2\_DR
5. ***USART2\_CR1***



L’USART2\_CR1 è un registro che presenta diverse funzionalità. Viene utilizzato nel progetto per: abilitare l’USART2 (bit 13 UE), abilitare la trasmissione e la ricezione (bit 2 RE e 3 TE) e per generare un interrupt USART ogni volta che ORE=1 o RXNE=1 nel registro USART\_SR (bit 5).

1. ***USART2\_BRR***



L’USART2\_BRR è un registro che viene utilizzato per settare il baud rate della comunicazione seriale della scheda di sviluppo. Il contenuto di questo registro è l’USART\_DIV che prevede una parte frazionaria e una mantissa. L’USART\_DIV è un divisore della frequenza di clock che decide la velocità della comunicazione secondo la seguente formula:



In cui OVER8 è in genere settato uguale a 0, Tx/Rx baud è il valore di baud rate che si vuole settare e fCK è la frequenza di conteggio del microcontrollore e da questa equazione è possibile ricavare tramite una semplice operazione di formula inversa l’USART\_DIV.

1. ***USART2\_SR***



L’USART2\_SR è un registro che tiene conto dello stato relativo alla periferica e viene utilizzato per verificare se è stato inviato o ricevuto un carattere.

1. ***USART2\_DR***



L’USART2\_DR è un registro dati della periferica e serve a contenere ciò che è stato immesso sulla seriale.

***NVIC***



L’NVIC (Nested Vectored Interrupt Controller) è un registro interno al Cortex (ovvero al processore del Microcontrollore) in cui vengono mappate, gestite ed abilitate o meno le sorgenti interne ed esterne di Interrupt. In questo caso sono stati abilitati il Timer2, Timer3 e Timer4 come sorgenti di interrupt (il Timer2 è quello che produce eventi di interrupt a maggiore frequenza) agendo sul registro NVIC\_ISER0 ed sono stati abilitati il Timer5 e l’USART2 sempre come sorgenti di interrupt agendo sul registro NVIC\_ISER1.

***MCUID***



L’MCUID1, MCUID2 e MCUID3 sono tre registri che sono stati dichiarati con la finalità di visualizzare l’ID del microcontrollore; in particolare, l’ID della scheda di sviluppo è una sequenza binaria a 96 bit contenuta in 3 registri ognuno da 32 bit (UID1, UID2, UID3).

***Timers Control Registers***

Immagine che contiene testo

Descrizione generata automaticamente

TIM2\_CR1, TIM3\_CR1, TIM4\_CR1 e TIM5\_CR1 sono dei registri utilizzati per controllare i vari timer utilizzati e per impostare le differenti modalità che la periferica offre all’utente; in particolare, nella progettazione è stato abilitato il conteggio settando in maniera opportuna i bit del registro.

***Timers Status Registers***

Immagine che contiene testo

Descrizione generata automaticamente

TIM2\_SR, TIM3\_SR, TIM4\_SR e TIM5\_SR sono dei registri di stato dei timer e tramite questi possiamo verificare se sono accaduti diversi eventi, come ad esempio la fine del conteggio.

***Timers Prescalers***

Immagine che contiene testo, interno

Descrizione generata automaticamente

TIM2\_PSC, TIM3\_PSC, TIM4\_PSC e TIM5\_PSC sono dei registri che consentono di settare il valore del divisore della frequenza di clock a cui conteranno i vari Timer e di gestire la frequenza di conteggio (all’interno di questi registri verrà impostato il valore per il quale andrà divisa la frequenza di clock, nel nostro caso il valore è 16MHz).

***Timers Auto Reload Register***

***Immagine che contiene testo

Descrizione generata automaticamente***

TIM2\_ARR, TIM3\_ARR, TIM4\_ARR e TIM5\_ARR sono dei registri in cui va inserito un valore intero a 32 bit che costituisce il numero di fronti di clock che verranno contati dal timer, terminato il conteggio il registro si auto-ricarica dello stesso valore.

***Timers DMA – Interrupt Enable Registers***

***Immagine che contiene testo

Descrizione generata automaticamente***

TIM2\_DIER, TIM3\_DIER, TIM 4\_DIER e TIM5\_DIER sono dei registri che hanno il compito di abilitare le differenti sorgenti di interrupt che la periferica dispone.

***Timers Event Generation Registers***

Immagine che contiene testo

Descrizione generata automaticamente

TIM2\_EGR, TIM3\_EGR, TIM4\_EGR e TIM5\_EGR sono dei registri che resettano il contatore e generano un aggiornamento dei registri.

**2.2 Descrizione delle funzioni sviluppate.**

*In questa sezione vengono descritte le funzioni utilizzate e il loro scopo.*

***Printstring***

Immagine che contiene testo

Descrizione generata automaticamente

La funzione printstring ha lo scopo di mettere a video tutto ciò che effettivamente si visualizza su Termite quando la scheda di sviluppo è collegata tramite cavo USB al PC. In queste 5 righe di codice viene eseguito un ciclo while in cui si entra se il valore della stringa (posto uguale al valore di USART2\_DR) a cui punta il puntatore è diverso dal carattere di fine stringa ‘\0’. Se la condizione è verificata si entra nel ciclo while in cui la flag della trasmissione completata di USART2 viene settata a 0 con un’operazione di ‘and’ ponendo a 1 tutti i e 32 i bit tranne il bit 6 e tramite l’utilizzo di un secondo ciclo while interno al primo si attende il termine della comunicazione del carattere tramite USART2, infatti il bit 6 del registro USART2\_SR diventa 1 quando la trasmissione è completa e la riga di codice aspetta proprio che tale bit diventi 1.

***Menù***

Immagine che contiene testo

Descrizione generata automaticamente

La funzione menù si serve della funzione printstring per mettere a video su Termite il menù principale da cui poi si possono scegliere le varie funzionalità che offre il software realizzato. Viene inoltre settata la static int submenù a 0, dove questa variabile statica è molto utile per gestire i vari “sottomenù” appunto che il programma presenta.

***SetDuty***

***Immagine che contiene testo

Descrizione generata automaticamente***

Immagine che contiene testo

Descrizione generata automaticamente

La funzione SetDuty ha lo scopo di permettere all’utente di settare il nuovo valore del Duty Cycle; quindi, la funzione ci restituisce il valore Duty inteso proprio come nuovo valore del Duty cycle scelto. Viene inizialmente dichiarata una variabile Ton e successivamente, entrando nello switch se il segnale è attivo (variabile statica SelectedSignal = 1), viene posta uguale al valore del registro TIM2\_ARRvalueON che sarebbe il valore di ON dell’Auto Reload Register del TIM2. Il valore di TIM2\_ARRvalueON viene posto uguale al valore del registro TIM2\_ARRvalueON più quello del registro TIM2\_ARRvalueOFF moltiplicato per il valore del duty diviso 100 perché il valore del duty è espresso in percentuale. Infine, il valore del TIM2\_ARRvalueOFF viene posto uguale a sé stesso più Ton meno TIM2\_ARRvalueON. Finito ciò la variabile statica SelectedSignal che tiene conto se il segnale è attivo o meno viene settata a 0 e poi si ritorna alla funzione menù. Lo stesso procedimento vale per gli altri tre casi.

***SetFreq***

***Immagine che contiene testo

Descrizione generata automaticamente***

***Immagine che contiene testo

Descrizione generata automaticamente***

La funzione SetFreq ha lo scopo di permettere all’utente di settare il nuovo valore della frequenza; quindi, la funzione ci restituisce il valore NewFreq inteso proprio come nuovo valore della frequenza scelta. Viene inizialmente dichiarata una variabile Duty e successivamente, entrando nello switch se il segnale è attivo (variabile statica SelectedSignal = 1), viene posta uguale al valore del registro TIM2\_ARRvalueON moltiplicato per 100 (perché il valore del Duty è espresso in percentuale) diviso il valore del registro TIM2\_ARRvalueON più il valore del registro TIM2\_ARRvalueOFF. Successivamente, TIM2\_ARRvalueON viene posto uguale al valore di Duty diviso il valore NewFreq il tutto moltiplicato per 10000 perché entrambi i valori sono espressi in percentuale. Poi, il valore di TIM2\_ARRvalueOFF viene posto uguale a 100 meno Duty ovvero il complementare del valore precedente diviso NewFreq e il tutto moltiplicato per 10000 sempre per lo stesso motivo. Finito ciò la variabile statica SelectedSignal che tiene conto se il segnale è attivo o meno viene settata a 0 e poi si ritorna alla funzione menù. Lo stesso procedimento vale per gli altri tre casi.

***IRQHandler***

Immagine che contiene testo, lettera

Descrizione generata automaticamente

Immagine che contiene testo, lettera

Descrizione generata automaticamente

La funzione IRQHandler viene eseguita nel momento in cui il timer 2 genera l’interrupt ossia quando termina il conteggio. All’interno del file “startup\_stm32f401xe.s” è specificato che la funzione da eseguire nel momento in cui la periferica precedentemente menzionata genera l’interrupt deve avere questo nome particolare. Utilizzare gli interrupt consente di ottimizzare il codice in quanto gli interrupt sono gestiti a livello hardware e non software. Per quanto riguarda l’USART2, questa funzione pone semplicemente il valore della variabile statica ReceivedCommand uguale al valore del registro USART2\_DR, mentre per quanto riguarda i Timer viene settata l’Interrupt Flag del Tim2 a 0, successivamente viene posto 0 al bit 0 del registro GPIOA\_ODR e poi si entra nel comando if – else in cui se l’operazione “and” tra il valore del registro GPIOA\_ODR e 1 è verificata allora viene posto il valore del registro TIM2\_ARR uguale al valore del registro TIM2\_ARRvalueON; se non è verificata la condizione allora viene posto il valore del registro TIM2\_ARR uguale al valore del registro TIM2\_ARRvalueOFF.

***ModifyBaud***

***Immagine che contiene testo

Descrizione generata automaticamente***

La funzione ModifyDuty setta la variabile statica submenù al valore 3 e si serve della funzione printstring per mettere a video su Termite ciò che è illustrato in figura.

***MCUIDread***

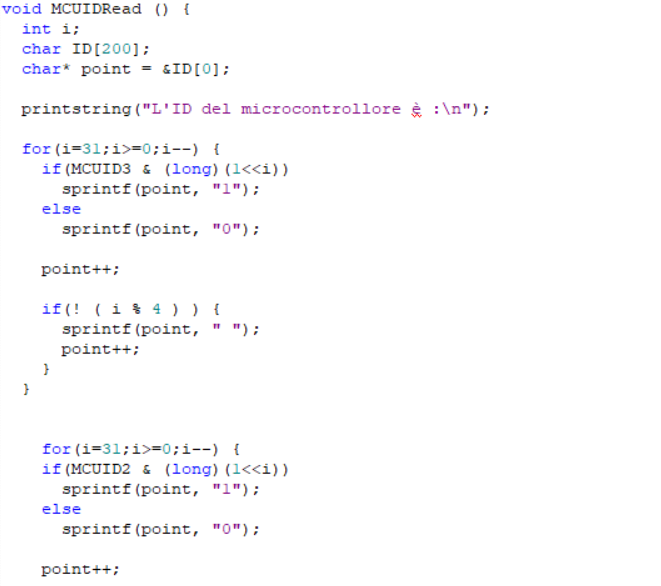


Immagine che contiene testo

Descrizione generata automaticamente

La funzione MCUIDread ha lo scopo di mettere a video su Termite l’ID del microcontrollore che è una sequenza binaria di 96 bit contenuta 3 registri ciascuno formato da 32 bit, ovvero UID1, UID2 e UID3. Inizialmente vengono dichiarati una variabile “i” utile per i passi successivi, una stringa di caratteri chiamata ID di dimensioni 200 caratteri per comodità e poi viene dichiarato un puntatore che “punta” al primo carattere della stringa chiamata ID. Dopo la funzione printstring viene eseguito un ciclo for al contrario altrimenti verrebbe letto il l’ultimo bit invece del primo e così via. All’interno del ciclo for viene esegui un comando if – else che “printa” 1 se l’operazione “and” tra il registro MCUID3 e la stringa con 1 al posto di “i” è verificata, altrimenti 0. Vi è infine un ulteriore comando if che serve a spaziare i bit “printati” ogni 4 bit. Le stesse operazioni vengono svolte per i registri MCUID2 e MCUID1; viene infine stampato l’ID completo e spaziato ogni 4 bit con la funzione printstring e si viene riportati alla funzione menù.

***Error***

***Immagine che contiene testo

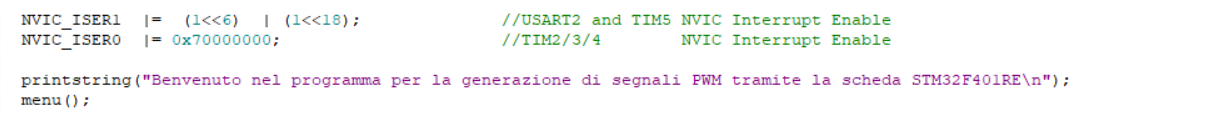
Descrizione generata automaticamente***

La funzione Error viene richiamata ogni qual volta viene digitato un carattere non permesso dal software sviluppato; viene così messo a video tramite la funzione printstring che è stato digitato un comando non valido e si viene riportati alla funzione menù.

***Main***

Immagine che contiene tavolo

Descrizione generata automaticamente



La funzione Main è la parte essenziale di tutto il programma in cui vengono abilitati i clock dell’USART e dei vari Timer, il clock della GPIOA, vengono settati i vari pin della scheda alcuni come Alternative Function (PA2 e PA3), in particolare come Alternative Function 07 e altri come Output (PA0, PA1, PA4 e PA5), viene abilitata l’USART2, l’Rx e il Tx e viene settato il baud rate a 9600 di default. Vengono poi abilitati gli interrupt sui vari Timer utilizzati tramite il registro TIMx\_DIER, vengono impostati i prescaler dei vari Timer a 16, vengono settati gli Auto Reload Register dei vari Timer a 0 e vengono attivati i contatori dei vari Timer, disabilitati gli interrupt update generation dei vari Timer, vengono abilitati gli NVIC interrupt dei vari Timer e dell’USART2 ed infine viene utilizzata la funzione printstring e si viene poi dopo riportati alla funzione menù.