PICSOFTWARE

Software per la programmazione dei sintetizzatori
ELCOM

adelmo
Adelmo.desantis@gmail.com

Prefazione

Il presente lavoro è il risultato della mia sperimentazione. E' frutto di un lavoro durato parecchi mesi e viene distribuito in modo gratuito e sotto licenza GNU Free Documentation License. Non viene data garanzia alcuna sulla effettiva funzionalità dei programmi, dei circuiti e delle soluzioni indicate in questo scritto. Non mi ritengo responsabile per danni diretti o a terzi derivanti dall'uso di quanto riportato in queste righe: la sicurezza prima di tutto!

Il mio contributo non vuole essere solamente di tipo tecnico. Vorrei che queste pagine contribuissero a diffondere un metodo di lavoro, basato sulla ampia condivisione dei risultati. Il progetto è messo a disposizione di tutti non solo per mera documentazione, ma per spingere gli interessati a "fare da soli", realizzando le board, modificando il codice e cercando di migliorarlo. Spero che tutto il tempo investito nel progetto non si riduca ad una mera realizzazione in serie delle schede (cosa nella quale non posso investire risorse). Pertanto leggete, chiedete lumi sulle parti meno comprensibili e poi cercate di realizzare il progetto. Sarà un ottimo modo per imparare qualche cosa di nuovo e contribuire allo sviluppo di nuove funzioni.

Versioni

Il presente documento è dinamico. Saranno aggiunte informazioni sulla base delle sperimentazioni che vengono effettuate o dei contributi ricevuti. La versione attuale del documento è la seguente:

Versione 1.0 Data di rilascio: 14 – Dicembre – 2015 Hardware di riferimento: V3.1

Changelog.

Versione 1.0
Release inziale del documento.

Sommario

Introduzione

Il software presentato è concettualmente e tecnicamente molto semplice. L'efficienza del programma è migliorabile sotto molti aspetti, purtroppo la mia limitata esperienza nella programmazione in C non mi consente una scrittura più compatta e più efficace.

I principali compiti del microcontrollore (e quindi del firmware) sono:

- Gestione della interfaccia utente;
- Programmazione del sintetizzatore;
- Programmazione del Si570.

Interfaccia Utente

L'interazione con l'utente è gestita attraverso una coppia di LED e un DIP switch a 9 posizioni.

La scheda è corredata da due indicatori LED. Chiameremo LED1 quello in alto, verso i connettori di alimentazione e LED2 quello in basso. I due indicatori vengono utilizzati per informare l'utente sullo stato operativo del sistema e sul risultato della selezione dei DIP Switch.

All'accensione della board, dopo alcuni istanti si illumina il LED2. Se è stato attivato il SI570, il LED1 lampeggia 3 volte. Quindi il LED2 si spegne. Dopo lo spegnimento del LED2 il LED1 lampeggia: il numero dei lampeggi riflette la scelta dell'utente del modello del sintetizzatore:

Numero lampeggio LED1	Selezione
1	DFS-1301 Rev.C
2	DFS-1301 Rev. D
3	Non Utilizzata
4	DFS-1201 Rev. D
5	DFS-1101 Rev. C
6	ILCDFSL-1201
7	ILCDFSL-1301 Versione I4SBX
8	ILCDFSL-1301
9	Test Sistema
10	Anomalia

Alla fine del lampeggio del LED1, il LED2 lampeggia in modo da indicare il numero della frequenza scelta per il particolare dispositivo. In caso di anomalia LED2 lampeggia 10 volte.

Al termine del processo di programmazione, sia il LED1 che il LED2 rimangono accesi.

Questa particolare codifica, consente di verificare immediatamente eventuali anomalie nel funzionamento del sistema

Programmazione Sintetizzatore

Il software supporta diversi sintetizzatori e, per ognuno di essi, un nutrito set di frequenze. La scelta del modello dell'oscillatore e della frequenza, avviene attraverso i DIP switch descritti nella sezione precedente.

Il primo passo è quello della lettura della configurazione del sistema da realizzare. Una volta determinato modello e frequenza viene richiamata la funzione principale "program" che agisce come una look-up table sulla base dei valori che le vengono passati come argomento. La funzione "program" richiama quindi una serie di funzioni, dedicate, che programmano il PLL con i corretti valori dei divisori.

I commenti presenti nel codice consentono di apprezzare i valori che vengono utilizzati nella programmazione ed il punto in cui entrano in gioco.

Programmazione Si570

La presenza di un ulteriore oscillatore a bordo della scheda, apre la possibilità a notevoli sperimentazioni. La gestione di questo dispositivo è piuttosto complessa e coinvolge diverse funzioni, oltre a richiedere il pieno supporto del bus I2C.

- Read_startup;
- Read_i2c;
- Write_i2c;
- Dco_status;
- FreqProg;
- tx_debug;
    - tx_bit1d
    - tx_bit0d

Read_i2c è una funzione molto semplice che implementa la lettura di un byte dal bus i2c. I dati sono messi nella variabile dataout di tipo unsigned int.

Write_i2c è una funzione che accetta in ingresso l'indirizzo del dispositivo, l'indirizzo del registro ed i dati da scrivere. Scrive un byte nel dispositivo all'indirizzo indicato.

Read_startup è molto importante in quanto effettua il reset del dispositivo e la lettura dei registri di configurazione originale del Si570. Questo consente di calcolare il valore di FXTAL, necessario per la corretta impostazione della frequenza di lavoro del dispositivo. Si tratta di una funzione molto onerosa in termini di codice in quanto lavora con float e double. Alla base del funzionamento della funzione è la popolazione dell'array REG[] che contiene il dump dei registri di lavoro del Si570. Sulla base di questi valori viene poi effettuato il calcolo della FXTAL.

Dco_status è una funzione molto semplice che effettua il freeze e unfreeze del dispositivo Si570. Viene utilizzata nelle fasi di programmazione della nuova frequenza.

Freq_Prog è la funzione più importante per la gestione di Si570. Accetta in ingresso il valore desiderato della frequenza di uscita (FLOAT) ed effettua il calcolo dei valori dei registri necessari per realizzarla. I registri sono calcolati utilizzando variabili di tipo double e float, in modo da non perdere troppa risoluzione. Considerando il numero di moltiplicazioni coinvolte, la funzione è molto onerosa per quanto riguarda l'implementazione in linguaggio assembly.

Tx_debug è una funzione ausiliaria che invia i dati di un vettore attraverso l'interfaccia di debug. Questa consiste di due PIN che realizzano una connessione seriale sincrona. Posizionandosi sui pin con un oscilloscopio è possibile verificare la sequenza di bit inviata. Richiama due funzioni ausiliarie: tx_bit1d e tx_bit0d.

Funzioni – Programmazione Sintetizzatori Elcom

Elenco delle funzioni e delle loro principali caratteristiche.

delay_long(usec)

Descrizione
Funzione per la generazione di ritardi utilizzando i timer del microcontrollore

Valori
0x00 – 82 us
0x01 – 164us
0x02 – 10ms
0x03 – 200ms
0x04 – 1s
0x05 – 6ms
0x06 – 100us

initHardware()
Descrizione
Attiva e configure l'hardware del microncontrollore.

Tx_bit1() – tx_bit0()
Sono due funzioni che realizzano l'interfaccia bit-bang del dispositivo.

set_enable_dfs(value)
Descrizione
Imposta il valore logico del pin ENABLE.

Valori
0x00 – Enable = 0;
0x01 – Enable = 1;
0x02 - Enable = 1 -> wait 100us -> Enable=0;

tx_status_dfs(model_select)
Descrizione
Programma il valore dello "status register" del PLL.

Valori
0x01 – DFS-1301 Rev. C
0x02 – DFS-1301 Rev. D
0x03 – Non utilizzato
0x04 – DFS-1201 Rev. D
0x05 – DFS-1101 Rev. C

tx_na_dfs(value_a, value_n_high, value_n_low)
Descrizione
Programma il valore del registro n/a del PLL. Accetta in ingresso il valore di a e il valore di n diviso su due byte in modo da potere trasmettere anche valori di N superiori a 256.

Valori
Valori numerici

tx_r_dfs(model_select,value)
Descrizione
Programma il valore del registro R per potere determinare lo step minimo possibile per l'oscillatore.

Valori
model_select:
0x01 – 0x02 DFS-1301 Rev.C e Rev.D
value:

0x01 – R=8;
0x02 – R=40;

model_select:0x04 DFS-1201 Rev.D
value:
0x01 – R=8;
0x02 – R=40;
0x03 – R=12;
model_select:0x05 DFS-1101 Rev.C
value:
0x01 – R=37;
0x02 – R=185;

initial_setup_dfs(model_select)
Descrizione
Programma il registro status del PLL. Utilizzata in fase di prima programmazione.

Valori
0x01 – DFS-1301 Rev.C
0x02 – DFS-1301 Rev.D
0x03 – Non utilizzato
0x04 – DFS-1201 Rev.D

Initial_values_dfs(model_select)
Descrizione

Effettua la prima programmazione del PLL, impostando la frequenza di default di sistema. Viene utilizzata in fase di avvio del PLL.

Valori
0x01 – DFS-1301 Rev.C
0x02 – DFS-1301 Rev.D
0x03 – Non usato
0x04 – DFS-1201 Rev.D

program(model_select, value)
Descrizione
Funzione che effettua la vera e propria programmazione del PLL. Per ogni modello di oscillatore sono possibili diverse frequenze pre-calcolate.

Valori
model_select
0x01 – DFS-1301 Rev.C
0x02 – DFS-1301 Rev.D
0x03 – Non Usato
0x04 – DFS-1201 Rev.D
0x05 – DFS-1101 Rev. C
0x0A – ILCDFSL-1201
0X0B – ILCDFSL-1301 I4SBX
0x0C – ILCDFSL-1301 Generico

	Value							
Model_sel	1	2	3	4	5	6	7	8
0x01	12672	12648	12600	12456	12650	13200	13000	13000
0x02	1	2	3	4	5	6	7	8
	12672	12648	12600	12456	12650	13200	13000	13000
0x03	--	--	--	--	--	--	--	--
0x04	1	2	3	4	5	6	7	8
	11772	11736	11414	11448	12024	11952	11808	11376
	9	10						
	11200	12000						
0x05	1	2	3	4	5	6		
	10512	10800	10575	11400	11000	10500		
0x0A	1	2	3	4	5	6	7	8
	11772	11736	11414	11448	12024	11952	11808	11376
	9	10	11					
	11200	12000	12000					
0x0B	1	2	3	4	5	6	7	
	10318	9984	10368,5	10368,4	10368,3	10368,2	10224	
0X0c	1	2	3	4	5	6	7	8
	12672	12648	12600	12456	12650	13000	13000	13200

test_dfs(model_value)

Descrizione

Funzione ausiliaria che programma una serie di frequenze sul PLL per scopo di verifica di funzionamento. Ogni frequenza rimane attiva per 5 secondi circa.

Valori

0x01 – DFS-1301 Rev.C
0x02 – DFS-1301 Rev. D
0x03 – Non usato
0x04 – DFS-1201 Rev.D
0x05 – DFS-1101
0x0A – ILCDFSL-1201
0x0B – ILCDFSL-1301 I4SBX
0x0C – ILCDSFL-1301 Generico

Frequenze Programmate:

Model_sel	1	2	3	4	5	6	7	8
0x01	12650	13200	13000 8	13000 40	12672	12648	12600	12456
0x02	12650	13200	13000 8	13000 40	12672	12648	12600	12456
0x03	--	--	--	--	--	--	--	--
0x04	12024	11952	11808	11376	11772	11736	11200	12000
	11414	11448						
0x05	10224	10512	10800	11664	10575	11400	11000	
0x0A	11772	11736	11414	11448	12024	11952	11808	11376
	11200	12000	12000					
0x0B	10318	9984	10368,5	10368,4	10368,3	10368,2	10224	
0x0C	12672	12648	12600	12456	12650	13000	13000	13200

Ilcfdfsl(model, rfr_r,rfr_mod,rfn_int,rfn_frac)

Descrizione

Funzione utilizzata per la programmazione dei sintetizzatori basati su AD4252. Muove dall'assunto che molti dei registri utilizzati nella programmazione non cambiano e sono quindi definiti in modo statico. Gli unici elementi che debbono essere modificati sono rfr (valore del registro r della sezione rf) ed rfn (valore del registro n della sezione rf). La loro popolazione viene eseguita facendo scorrere il valore inserito da utente su un AND e scrivendo il risultato in un vettore. Terminata questa fase, viene programmato il dispositivo con l'aiuto della PLL_program_dfs.

Valori

model: discrimina tra 1201 e 1301;
rfr_r: valore del divisore R in hex;
rfr_mod: valore del divisore mod in hex;
rfn_int: valore del divisore int in hex;
rfn_frac: valore del divisore frac in hex.

*PLL_program_dfs(number, *value, direction)*

Descrizione

Funzione che riceve in ingresso il numero dei bit da trasmettere, il vettore che li contiene ed una indicazione di direzione. Richieama le funzioni tx_bit1() e tx_bit0() per creare la bit-bang interface.

Valori:

number: numero dei bit da trasmetter;
*value: puntatore al vettore che contiene il registro da trasmettere;
direction: indicatore di direzione di trasmissione: LSB first (0) MSB first(1).

Blink_led (nblink,speed) e blink_led2(nblink, speed)

Funzioni accessorie chef anno lampeggiare il led1 o il led2 per nblink volte ad una velocità "speed".

Wait_5s()

Ritardo lungo.

Led_numbers(value)

Funzione utilizzata per il debug. Riproduce il valore immesso nella variabile value accendendo i led1 e led2.

Schemi a blocchi di alcune funzioni

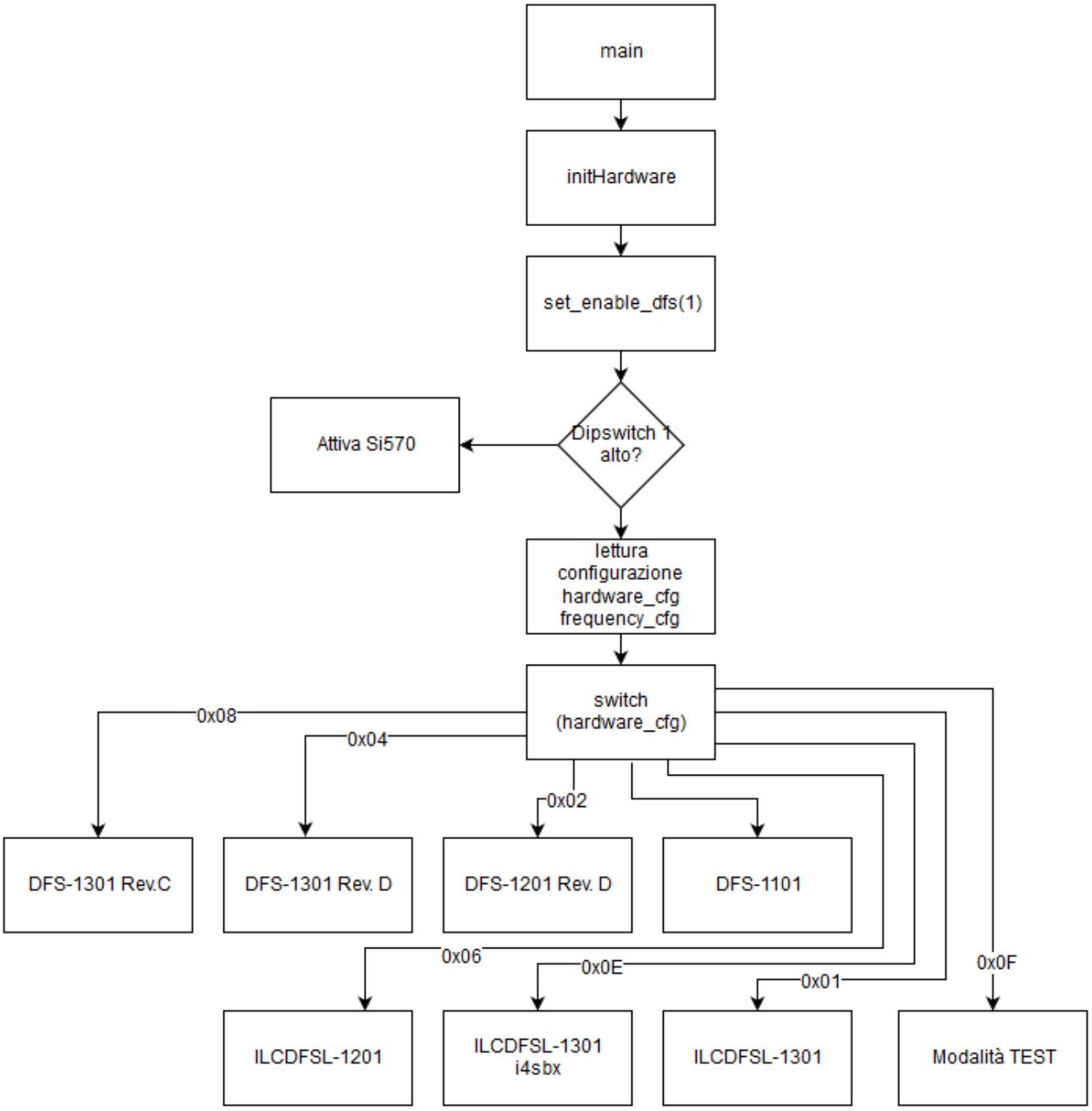

Figura 1 - Funzione main

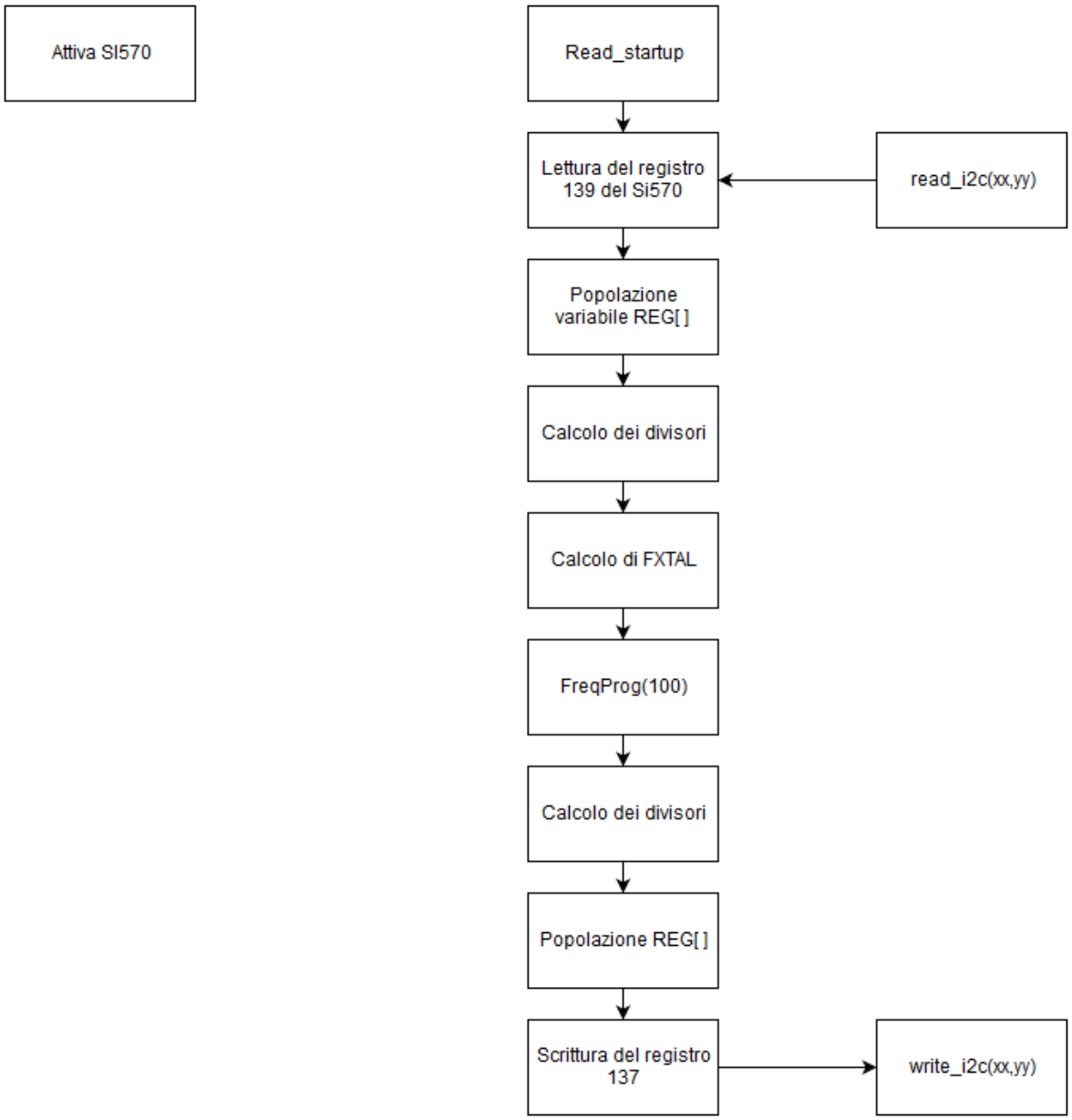

Figura 2 - Programmazione di Si570

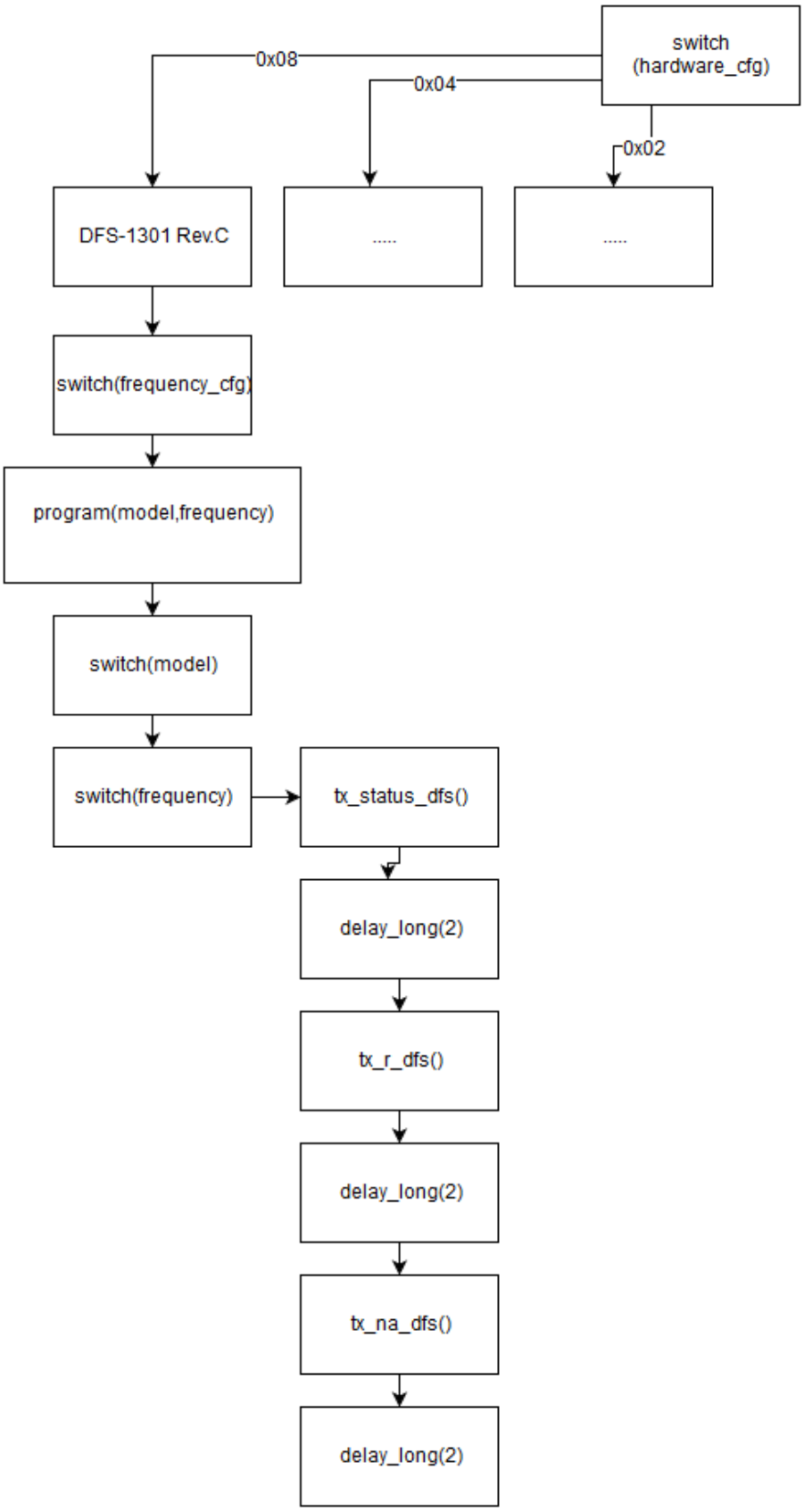

Figura 3 - Programmazione del PLL

Figura 4 - Funzioni ausiliarie alla programmazione del PLL

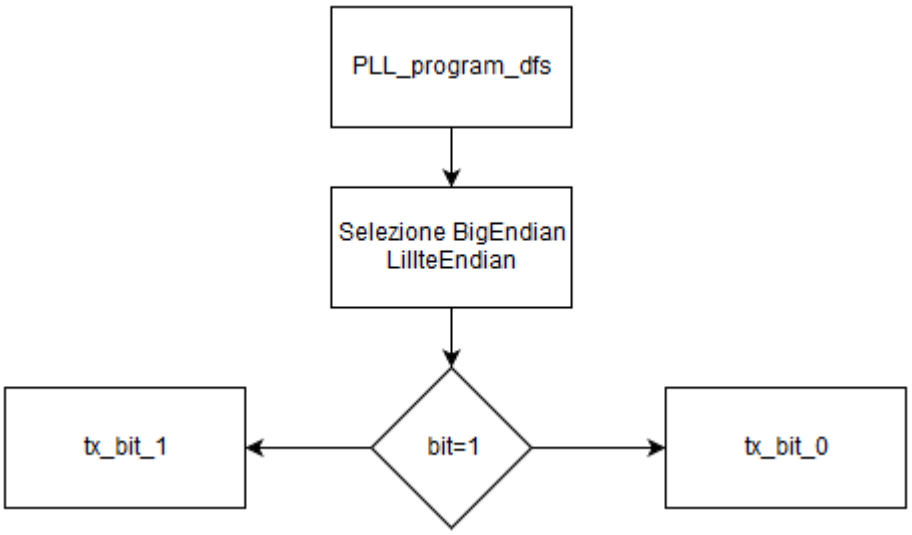

Figura 5 - Funzione bit-bang

Sistema di sviluppo

Il firmware è stato sviluppato utilizzando MicroChip MPLABX e compilatori XC8. Per consentire il corretto funzionamento del codice, soprattutto nella parte che riguarda il pilotaggio del SI570 è necessario impostare il linker in modo da potere sfruttare 32 bit per i tipi double ed float.

Nella impostazione del compilatore è importante che nella sezione “XC8 global options – XC8 Linker – Memory Model” sia scelto il valore di 32bit sia per il “Size Of Double” che per il “Size of Float”, come mostrato in figura.