Kirkegaard Winkler (Kirkegaard)

LO STRAPPO … CONTINUA

2 September 2011

Cara Roberta Sparrow, ho letto a fondo il suo libro e ci sono molte cose che vorrei chiederle. A volte ho paura di quello che lei potrebbe dirmi. E a volte ho paura che lei mi dica che non è tutto frutto della fantasia. Posso solo sperare che la risposta mi arrivi nel sonno. E spero anche, quando il mondo finirà, di poter tirare un sospiro di sollievo, perché ci sarà tanto da contemplare avidamente.

[Lettera di **Donnie Darko** a Roberta Sparrow]

Cosa significa lavorare con un micro senza interrupt? Se devo essere sincero all'inizio ero pure contento perché gli interrput, le interruzioni, sono l'elemento di disturbo nel flusso del programma.
Il firmware lo vedevo ancora come se fosse un programma basic monolitico e gli interrupt erano la fastidiosa eccezione. Ma il pic aveva altre sgradite sorprese che lo distinguevano dagli altri micro. La struttura interna è di **tipo Harvard modificato** e a differenza della architettura **Von Neumann** questo micro ha i **bus** della memoria **dati** e quello della memoria **programma** separati.

Questa differenza ha permesso, alla Microchip, di ridimensionare la memoria programma differenziandola da quella dati. Così la struttura dati della memoria programma fu portata a 12 bit. Cosa significava questo?
In primo luogo che diminuirono drasticamente gli opcode assembly tipicamente molto numerosi nei micro concorrenti e riuscendo con un singola , ma efficentissima, istruzione ad effettuare tutte le funzioni base del micro.(intendendo quindi le operazioni di spostamento fra registri, le operazioni logiche i test dei bit e così via…). In questo modo riuscirono a creare una tabella di opcode di solo 33 istruzioni mentre, che ne so, lo Z80 ne aveva più di 150 . In secondo luogo i 2 kb di programma del pic corrispondevano in reatà a più di 3 k di memoria degli altri micro. Quando la Microchip incominciò a pubblicare le analisi sull'efficienza di sviluppo del codice firmware, fece non poco girare les pelotas (scusate per lo spagnolo…) agli altri concorrenti (l'Atmel su tutti), che a ruota risposero con altre application note sui parametri effettivi che si dovevano considerare per ritenere il codice davvero efficiente, nel tentativo di dimostrare che il loro micro era il migliore in commercio. Sembrava di assistere alle pubblicità americane della pepsi e della coca cola…
La doppia pipeline del PIC permetteva di eseguire un'istruzione ad ogni ciclo macchina per tutte e trentatre (33) istruzioni. L'unica eccezione l'avevano i salti condizionati che necessitavano di un ulteriore ciclo macchina per eseguire il salto.

Una caratteristica interessante che sembrava una stranezza, ma che poi verificai direttamente con questo progetto, la includo così come l'hanno pubblicata nel datasheet del micro.

```
"The PIC16C5X has a highly orthogonal
(symmetrical) instruction set that makes it possible to
carry out any operation on any register using any
addressing mode. This symmetrical nature and lack of
'special optimal situations' make programming with the
PIC16C5X simple yet efficient. In addition, the learning
curve is reduced significantly."
```

Si, la curva di apprendimento si riduceva ed era vero … ma non gli improperi e le pastiglie di Malox che invece aumentarono esponenzialmente.
Uno dei problemi più rilevanti, era la struttura a pagine sia della memoria dati che della memoria programma.
Quelle maledette pagine rendevano difficoltosa ogni modifica sul codice e tutte le revisioni diventavano una vera tortura.
Una volta consolidato il firmware, rimanevano da mettere a posto solo gli aspetti secondari e la curva dei bachi diminuisce sino a che non si azzera. La curva dei bachi, peraltro si azzera ma non del tutto, perchè quelle maledette cinghie slittavano, ed era maledettamente vero quello che l'infingardo tecnico rilevò.
Predisposi due lavatrici in modo da vedere il cestello girare, una con la cinghia nuova e una con la cinghia vecchia.
Non potei fare altro che confermare quanto denunciato, almeno una volta su 5 partenze, si sentiva quell'inesorabile suono sibilante della cinghia (nuova) che slittava sull'albero motore, con tanto di evidente vibrazione visiva…
A questo punto il controllo PID entrò direttamente fra gli imputati, così come tutta la parte di controllo del triac, ma quando accadono questi eventi non ripetibili, le cose sono molto più complicate di quello che sembrano, e coinvolgono tutto il sistema embedded.
L'oscilloscopio è stato sempre di valido supporto nelle ricerche di bug e di test dei segnali.
Per questo tipo di scheda era, diciamo, un po' problematico il suo utilizzo, perché questa elettronica sviluppata nel settore consumer non ha sezioni di alimentazione con trasformatore, e quindi non è separata galvanicamente, ma lavora direttamente sulla 230 volt. (si … ma non ditelo a quelli della usl…).
Per poter lavorare con un oscilloscopio sulla scheda, abbiamo cosi disaccoppiato l'oscilloscopio con un trasformatore 220/220, e questo mi avrebbe permesso di fare le misure necessarie in tutta sicurezza sulla scheda…. (ovviamente la sicurezza era solo per l'oscilloscopio e non per il sottoscritto….).
Lo strappo avveniva proprio alla partenza del motore, che è un momento delicato per l'attivazione.

In questa fase il PID viene inibito e l'attivazione del motore deve essere fatta accelerando con una certa pendenza.
Se questa accelerazione è troppo lenta o troppo veloce, il PID si attiva in momenti anomali pregiudicando il loop di stabilizzazione. Anche riuscendo a controllare la derivata della accelerazione, questi inneschi anomali sono causati in particolar modo dal fatto che la tachimetrica in questa fase è molto instabile.
Il segnale per la bassa velocità ha una ampiezza molto piccola che è vicina alle soglie del pin di ingresso del micro, e così spesso si rilevano fronti spuri che non esistono. Questo rende particolarmente difficile per il processore riconoscere il momento per passare dalla rampa manuale al controllo con il PID. (e si … è dura la vita…)
Con una sonda mi misi sul triac cercando di visualizzare lo strappo, mentre l'altra la misi in uscita dall'operazionale per visualizzare invece lo stato della tachimetrica.
La cosa era preoccupante perchè l'attivazione del triac era molto lineare e la derivata dell'accelerazione era perfettamente controllata. Provai a modificarne la pendenza, ma non c'era verso; la cinghia continuava a strappare con la stessa frequenza.
In realtà quello che ritenevo sistematico era molto variabile e la frequenza presunta non era uno su cinque come credevo.. ma variava.
Provai a implementare qualche strano algoritmo con innesco a scalino ma niente da fare un altro giorno era passato e la soluzione la vedevo lontana...
molto lontana...