# Una Strategia Ottimizzata per il Progetto di Sistemi Embedded

*By (Jon Pearson, Cypress Semiconductor Corp.)*

## *Sommario*

In presenza di problema complesso, la sua suddivisione in problemi specifici più semplici che vengono affrontati con i tool più idonei è una metodologia valida, applicabile anche nel caso dei sistemi embedded.

## *Introduzione*

Benchè gli utenti non ne avvertano generalmente la presenza, i controllori embedded sono dappertutto. I telefoni cellulari, per esempio, contengono fino a una dozzina di dispositivi embedded che controllano qualsiasi cosa, dalle trasmissioni radio al display che svolge la funzione di interfaccia utente, dalla tastiera all'intensità della retroilluminazione a LED. Questo fatto riflette una tendenza abbastanza ovvia: in presenza di un problema complesso, si procede i una suddivisione in problemi specifici più semplici che vengono quindi affrontati con i tool più idonei. Un procedimento di questo tipo si può applicare al sistema embedded più banale, con il risultato di abbreviare il ciclo di progetto, aumentare la flessibilità incrementare il livello di controllo. L'elemento chiave per garantire un esito positivo è l'utilizzo di una strategia di comunicazione valida in linea generale che fa apparire omogenee al mondo esterno le funzioni locali e remote.

## *Master-slave, host-client, hub-nodo: un concetto univoco*

Qualunque espressione si voglia usare, il concetto è il medesimo: concentrare il processo decisionale (master) e distribuire le azioni da svolgere (slave). Un cellulare, per esempio, dispone di un processore master che decide quello che deve essere presentato sullo schermo e quali periferiche, come ad esempio il controllore del display LCD o il transceiver radio, sono necessarie per lo svolgimento dell'azione. E' estremamente improbabile che il processore master controlli direttamente i singoli pixel del display o la codifica e decodifica del segnale radio. Il master comunica al dispositivo slave del display quello che deve visualizzare e questo deciderà la modalità da seguire; allo stesso modo il master fornisce al transceiver (che agisce da slave) il segnale audio da codificare e questi renderà disponibile il segnale audio che ha decodificato.

In altri sistemi embedded la suddivisione non è così ovvia, anche se le motivazioni e i benefici sono i medesimi. Se il dispositivo master principale può comunicare sia con i dispositivi slave remoti sia con quelli locali, il controllo può mantenersi coerente in tutto il sistema. Per un sistema diviso oppure distribuito, è indispensabile che le interfacce vengano specificate in modo esatto. Il fatto che tutte le autovetture abbiano il pedale dell'acceleratore a destra e quello del freno a sinistra permette a chiunque di affittare una macchina in ogni parte del globo e di guidarla senza problemi.

Sebbene per i microcontrollori siano disponibili parecchi metodi di comunicazione standard, tre sono più frequentemente utilizzati nei sistemi embedded di tipo master/slave: seriale RS/232, SPI e I2C. I dispositivi slave che adottano queste interfacce di comunicazione sono liberamente disponibili sul mercato. Alcuni esempi di dispositivi a più basso livello sono convertitori A/D e D/A, EEPROM seriali e varie tipologie di I/O digitali. Componenti per il monitoraggio e la messa in sequenza della tensione e controllori di ventole ad nello chiuso sono esempi di dispositivi di livello più elevato. Da soli, questi dispositivi fanno poco o niente: una volta integrati in un sistema con un dispositivo master che comunica mediante un bus standard, questi slave diventano parte di un sistema di più ampie dimensioni e, grazie al controllo attivo del master esercitato attraverso il bus di comunicazione, rendono possibile l'implementazione di funzionalità più avanzate.

## *I2C: uno standard "de facto"*

Nel momento in cui si esamina l'offerta di dispositivi slave e di microcontrollori che potrebbero essere adoperati come master o slave personalizzati, balza immediatamente all'occhio la vasta diffusione del bus I2C (Inter-Integrated Circuit). Due essenzialmente le ragioni della sua popolarità: il bassissimo costo (per realizzare il bus sono sufficienti due fili/pin e due resistori di pull-up) e la semplicità d'uso. Esistono molto specifiche ragioni per scegliere un bus al posto di un altro ma, in genere, il bus I2C a 400 MHz rappresenta la scelta migliore per un sistema embedded distribuito.

Il principale vantaggio di un approccio che prevede la suddivisione in diversi sottosistemi è derivato dall'ovvia considerazione che i problemi semplici sono più facilmente risolvibili rispetto a quelli complessi. Inoltre, separando fisicamente i dispositivi, si riducono i problemi di accoppiamento con conseguente aumento dell'affidabilità del sistema. Mediante un'idonea progettazione (assegnazione delle funzioni e definizione delle interfacce) è possibile evitare le modalità di guasto più diffuse. Con l'esperienza, la suddivisione permetterà di ottenere una maggior numero di componenti di progetto riutilizzabili, così da evitare di ripartire da zero durante lo sviluppo del progetto successivo (o ancora meglio di concentrare l'attenzione sull'apporto di eventuali migliorie).

## *Una guida pratica*

L'approccio utilizzato prevede la presenza di un bus di comunicazione e dà la possibilità di utilizzare un bus diverso da I2C, in quanto valido indipendentemente dal bus selezionato. E' necessario delineare con chiarezza una struttura di tipo master/slave e in alcuni casi il bus influenzerà alcune delle specifiche, come ad esempio il punto di inizio di una comunicazione. In questo esempio si farà riferimento al bus I2C a causa della sua diffusione e del fatto che i tool esterni necessari per il collaudo e il debug sono poco costosi o di semplice realizzazione.
Applicando il concetto master/slave, ogni progetto embedded può essere gestito in modo simile. In primo luogo si determina quali decisioni devono essere prese a livello centrale e si assegnano all'host, quindi si distribuiscono le azioni agli slave. La maestria sta nel delineare i confini. Una strategia efficace prevede che il master non debba restare in attesa per uno slave per qualsiasi cosa e se uno slave richiede l'intervento del master, deve procedere a una chiamata. Questo approccio prevede in alcuni casi la presenza di una terza linea, ovvero quella di interrupt, in modo che lo slave possa "chiamare" il master.

## *La strategia è il primo passo*

Prima di iniziare il progetto è necessario individuare una strategia che sia la più semplice possibile. In ogni caso essa deve essere annotata, perché ciascun membro coinvolto possa seguire la medesima strategia e, nel caso di eventuali modifiche, queste possono essere condivise con estrema semplicità. Una strategia chiara seguita da parecchi gruppi che lavorano insieme oppure in maniera indipendente, permette la condivisione dei progetti e comporta la riduzione dei cicli di sviluppo a fronte di un maggior grado di sicurezza.
Poiché il progetto si riferisce a un sistema composto da parecchie parti, su alcune di esse il controllo può non essere completo (come accade nel caso dei componenti standard): per tale motivo è necessario prendere in considerazione i vincoli specifici dei dispositivi. Grazie all'esperienza, una strategia chiaramente delineata rappresenta un valido ausilio sia nella progettazione della parte custom del sistema sia nella scelta dei dispositivi standard, in modo da garantire una maggiore flessibilità e il ricorso a un minor numero di compromessi nelle fasi finali del progetto.

## *Il compito degli slave*

L'obbiettivo è avere un dispositivo master in un sistema embedded in grado di fornire le direttive agli slave anche se l'esecuzione delle misure, l'analisi dei dati e le azioni intraprese saranno in linea generale processi autonomi. Tutto ciò, abbinato alla strategia menzionata sopra (il master non deve mai aspettare, lo slave esegue le chiamate solo quando necessario), permette di valutare l'efficienza della progettazione distribuita. Ogni dispositivo esegue le operazioni che sa assolvere al meglio e solo quando richiesto.

## *Non solo I2C*

I concetti discussi non si riferiscono a nessun tipo di bus di comunicazione in particolare. Quello che viene richiesto è la definizione del protocollo che soddisfi una particolare strategia (che può comprendere la minimizzazione del tempo di trasmissione nel caso di un sistema wireless alimentato a batteria oppure la presenza di funzioni affidabili per il rilevamento e la correzione degli errori nel caso di funzionamento in ambienti industriali gravosi). Il master potrebbe essere chiamato hub e non essere l'iniziatore, mentre lo slave può essere denominato nodo e avere un tempo di risposta pianificato, ma i concetti fin qui espressi restano validi. In ogni caso è necessaria una chiara definizione delle interfacce così da permettere a coloro che si occupano dello sviluppo di progetti futuri di modificare la realizzazione senza influenzare i componenti di livello superiore.

## *Controllore del pannello frontale con expander di I/O I2C*

La più semplice traduzione in pratica dei concetti espressi nell'articolo è rappresentata da un sistema preposto al controllo del pannello frontale di una parte di apparecchiatura. Il pannello è composto da tasti, commutatori e LED. Per sviluppare il progetto di questo pannello, che è un sottosistema di un apparato più complesso, il master sarà il processore principale preposto alla gestione dell'intero apparato (solitamente un computer che opera in ambiente Windows o Linux), mentre per gli slave sarà possibile scegliere tra decine di expander (blocchi di espansione) di I/O collegati al bus I2C. Selezionando expander di I/O standard collegati al bus I2C che vengono configurati dal processore master in fase di accensione, gli ingressi di rilevamento dei pulsanti e dei commutatori e le uscite di stato dei LED sono connessi all'interno del processore master. Ciascun processore dotato di bus I2C può fungere da master e la configurazione dei pulsanti e dei LED può variare con facilità, così come i dispositivi slave specifici. I dispositivi sono configurati via software all'interno del master: nella figura 1 è riportato lo schema di questo semplice sistema. I circoli blue rappresentano gli ingressi dei pulsanti, mentre le forme rosse e verdi rappresentano i LED. Una domanda ovvia potrebbe essere legata all'opportunità di utilizzare un dispositivo di più ampie dimensioni invece di due più piccoli. Le motivazioni alla base della scelta fatta in questo contesto sono essenzialmente due:

da una parte è meglio isolare le funzioni di ingresso e di uscita e dall'altra è preferibile conferire modularità al progetto in maniera "intelligente", così da permettere l'apporto di modifiche al sottosistema di ingresso senza coinvolgere il sottosistema di uscita. L'hardware è dunque molto semplice e, poiché tutte le funzioni di controllo sono accentrate nel master, le variazioni sono semplici da apportare. Lo svantaggio di questo approccio è la limitatezza in termini di capacità. Infatti, cosa succederebbe se si decidesse di aggiungere un sensore per la luce ambiente che regola la luminosità dei LED? L'onere ricadrebbe sul master che dovrebbe "riunire" tutti gli slave e adattare tutte le modifiche di progetto, anche se la sua prospettiva nei riguardi del pannello frontale rimane sostanzialmente inalterata: il master deve solo sapere quali pulsanti vengono premuti e ordinare l'accensione o lo spegnimento dei LED.
Riandando con la memoria alla strategia delineata, è lecito chiedersi perché il progetto originale non riesce a soddisfarla. In primo luogo, tutta l'"intelligenza" è concentrata nel master e gli slave richiedono un'attenzione quasi costante.
Invece è possibile definire l'interfaccia del pannello frontale e lasciare i dettagli a un sistema distribuito di slave (Fig. 2) che necessitano di una minore interazione da parte del processore principale.
Rimane il dilemma circa la metodologia da seguire per generare il dispositivo custom senza incorrere in problemi di una certa entità. A tal proposito è possibile utilizzare un'implementazione collaudata dello slave I2C che definisca un'interfaccia e un protocollo stabili: il protocollo più comune è quello basato su registri adottato da moltissimi dispositivi slave I2C standard. Di seguito viene riportato un breve riassunto relativo ai dettagli di questo protocollo:
•Per il bus I2C, tutte le transazioni vengono avviate dal master;
•Ogni dispositivo slave ha un indirizzo I2C a 7 bit (l'LSB indica se la transazione è un'operazione di lettura o scrittura);
•Ogni dispositivo slave ha un registro degli indirizzi interno che mantiene un puntatore a una "tabella" interna di dati, comandi o stati;
•Ogni dispositivo slave definisce gli indirizzi dei suoi registri e delle loro funzioni, inclusa l'indicazione che si tratti di dispositivi di sola lettura o sola scrittura;
•Una transazione di scrittura è composta da un byte formato dall'indirizzo a 7 bit del dispositivo I2C e dal bit di scrittura, seguita da un byte che imposta il registro degli indirizzi interno: se nella transazione vi sono altri byte, essi saranno scritti dal dispositivo slave a partire dall'indirizzo interno appena impostato (dal secondo byte);
•Una transazione di lettura è composta da un byte formato dall'indirizzo a 7 bit del dispositivo I2C e dal bit di lettura: il dispositivo master sincronizza il numero di byte dallo slave prescelto e fornisce un segnale di stop per indicare la fine dell'operazione.

Come si può notare, uno slave I2C ricorda molto da vicino una RAM dual port ed è quasi altrettanto semplice da utilizzare.
A questo punto è necessario selezionare il dispositivo programmabile: si tratta di una scelta molto personale, anche se l'elemento da tenere nella massima considerazione è la capacità di soddisfare una pluralità di esigenze senza dove procedere a successivi processi di apprendimento. I dispositivi PSoC di Cypress Semiconductor, assimilabili per tantissimi versi a un microcontrollore, dispongono di tutte le caratteristiche necessarie per gestire un gran numero di slave I2C e rendono disponibile un tool di uso semplicissimo fa dell'aggiunta di funzionalità slave I2C un'operazione quasi banale.
In primo luogo si procede alla definizione di un dispositivo slave per l'ingresso del pulsante, creando un componente con tre pin di indirizzo e sette ingressi per il pulsante, configurato in modo tale da accettare un commutatore normalmente aperto che chiude l'alimentazione del sistema (pari a 5 V c.c.). Gli stati dei pulsanti saranno disponibili in un singolo byte accessibili da un master I2C.
In secondo luogo è necessario definire il dispositivo slave per il controllo dei LED, dando vita a un dispositivo con due pin di indirizzo che pilota 8 LED (con un corrente di 10 mA), quattro rossi e quattro verdi. Poi viene definito un singolo byte, chiamato "Comando", che agisce come ingresso di comando I2C per il controllo dei LED: i quattro bit inferiori sono responsabili dei LED rossi e i quattro superiori di quelli verdi.
A questo punto è possibile fermarsi, avendo fornito una versione personalizzata dei dispositivi standard, oppure proseguire con una definizione più dettagliata dell'interfaccia master/slave I2C custom, sfruttando le specifiche di progetto, per una personalizzazione più spinta del livello slave. Probabilmente risulterà utile l'aggiunta di una linea di uscita al dispositivo di ingresso del pulsante se si vuole tramutare il master da un dispositivo a interrogazione ciclica in uno funzionante mediante interrupt che viene avvisato, ad esempio, delle variazioni di stato di ogni singolo pulsante. Si tratta di migliorie che tendono a isolare il processore centrale dai dettagli di livello più basso e permettono di ottimizzare i singoli sottosistemi senza influenzare il sistema principale.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com