Sistemi di controllo trasmissione dati seriali: correzione di errori, implementazioni custom (2)

Dopo aver visto in esteso le metodologie più comuni nell'attuare le «reti broadcast», atte a controllare e pilotare device intelligenti, le loro caratteristiche più salienti ed i problemi connessi al loro uso, questa volta ci occuperemo dei controlli sugli errori, protocolli protetti ed implementazioni standard industriali

di Massimo Novelli

Rivelazione degli errori di trasmissione

Vi sono molte cause che possono indurre errori di trasmissione in una rete di comunicazione dati seriali ad alta velocità. «Spikes» elettrici, interni o esterni alla rete, che vengono interpretati come stati logici non voluti, cattive connessioni date dai cavi o dai loro terminali che inducono «bit drop-out» oppure, soprattutto nel caso di reti «party line», trasmettitori che siano attivi allo stesso tempo.

Le tecniche comunemente usate per testare gli errori sulle trasmissioni dati sono fondamentalmente tre e sono «Parity», «Checksum» e «Cyclic Redundancy Check». A seconda delle probabilità che errori avvengano più o meno frequentemente, esse potranno essere usate da sole o in combinazione tra loro.

La più semplice delle tecniche sarà quella di inserire un bit extra in ogni byte trasmesso; questo bit, chiamato di parità, è usato per settare il numero totale di «RF presenti nel flusso di bit affinché sia reso di quantità pari (even parity) oppure dispari (odd parity). Tipicamente poi se i dati sono trasmessi come veri byte binari (tutti gli otto bit sono significativi), il bit di parità sarà aggiunto immediatamente dopo il byte relativo, mentre se sarà trasmesso un dato ASCII esso sarà incluso come l'ottavo (più significativo) databit di ogni byte trasmesso.

L'uso dei bit di parità darà un buon risultato su errori di trasmissione a base «byte-by-byte», ma questo guadagno in verità è a scapito senz'altro della velocità intrinseca finale, a causa dell'aggiunta di esso ai byte trasmessi, mentre soffre dell'incapacità di rilevare errori che possano avvenire in due databit monitorati (o in ogni suo multiplo pari).

La correzione detta di «checksum», in sostanza, si potrà equiparare ad un byte determinato aggiungendo l'equivalente valore, senza il segno (unsigned), di ogni byte nel flusso di bit seriali. A differenza della tecnica di parità, il metodo checksum potrà rilevare «multiple-bit error» in un byte, mentre, comunque, errori rilevati in due o più byte potranno invalidare simile tecnica di correzione. Questa metodologia, specie se usata in congiunzione con la precedente, di norma riduce di molto la possibilità che errori multipli su bit diversi possano passare inosservati e non corretti.

Quando invece sarà imperativo che la trasmissione dati debba essere virtualmente esente da errori, si potrà quindi impiegare la tecnica detta di «Cycling Redundancy Check» (CRC). Tale metodo genererà una parola binaria (Bynary Word) in modalità polinominale anziché essere semplicemente aggiunta al flusso. La possibilità che un errore rimanga non scovato usando una simile tecnica è estremamente bassa ed ovviamente, a livello di implementazione, molto più complessa da attuare e da derivare. Ci si potrà quindi ritenere fortunati sapendo che è presente sul mercato tutta una serie di device IC pronti allo scopo e che in modo funzionale generano e verificano gli stadi CRC in ogni applicazione.

Protocolli di comunicazione

L'aspetto chiave di ogni sistema di comunicazione dati seriali è senz'altro lo sviluppare metodi per trasmettere informazioni significative da un punto ad un altro. Gli attuali protocolli di comunicazione determinano come le unità del sistema interagiranno con le altre unità ed attualmente non esiste, in ambito broadcast, uno stabile, completo set di protocolli che possa soddisfare l'industria in applicazioni data-link ad alta velocità. Ve ne sono comunque di diversi che specificano a grandi linee i formati di carattere generale, e come esempio di quello che dovrebbe essere considerato una base di discussione, possono essere prese in considerazione le convenzioni di standard che vanno sotto il nome di HDCL/SDCL.

Il cosiddetto HDCL (High-level Data link Control) e lo SDCL (Synchronous Data link Control) sono in sostanza due varianti della stessa cosa, per cui ci rife-

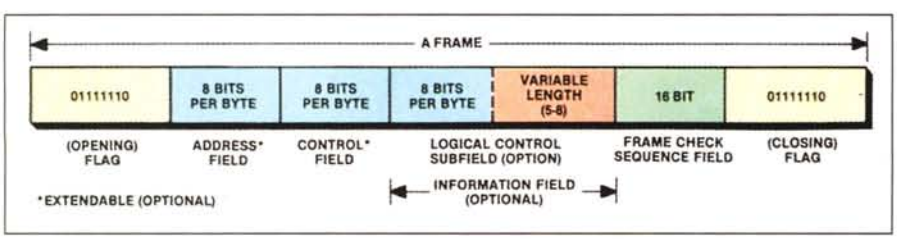


Figura 1 - Il formato del protocollo di comunicazione HDCL/SDCL (High level Data link Control/Synchronous Data link Control).

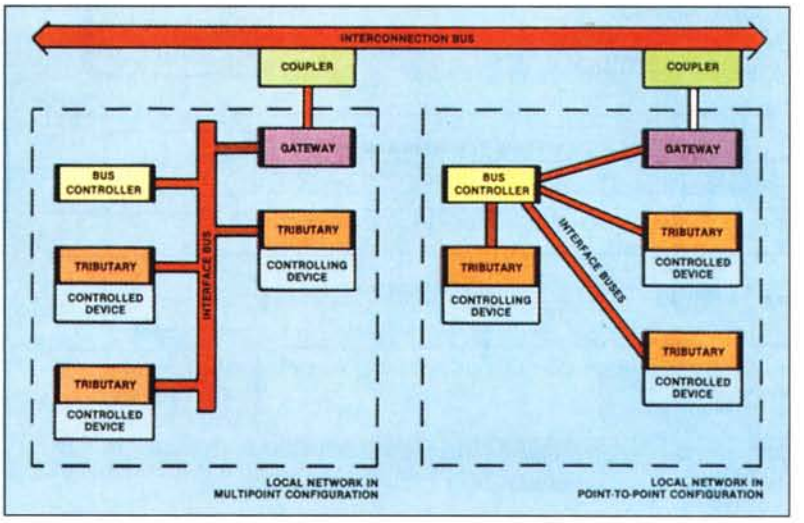


Figura 2 - Una tipica interconnessione ES-Bus; essa non specifica in standard gli accoppiatori o i device controllati. Solo gli elementi gateway e tributary sono soggetti alle specifiche.

riremo, in questa nostra discussione, al solo HDCL.

I protocolli HDCL specificano solo il formato dei dati del flusso seriale: non specificano caratteristiche elettriche o tecniche di codifica, mentre invece determinano perentoriamente che non siano adottate codifiche di tipo asincrono. Un tipico flusso di dati HDCL è mostrato in figura 1. Per indicare che il datastream è in corso, sarà trasmesso un byte di «flag» e questo byte ha un univoco formato binario di «01111110». Il secondo byte del flusso sarà l'indirizzamento al device con cui dialogare ed a seguire il byte usato per determinare il comando voluto dal dispositivo collegato, mentre i dati addizionali richiesti saranno trasmessi dopo il «command byte».

Completerà il pacchetto (chiamato frame) una word a 16 bit CRC, per la correzione di errori, mentre un byte di flag di chiusura completerà il tutto.

Il protocollo HDCL è senz'altro un buon esempio di come poter indirizzare, senza complesse manovre, e con tutte le caratteristiche richieste dall'industria, il controllo di device collegati in rete; in esso infatti si avranno indirizzamenti, comandi, azioni e rilevazione di errori in una unica sequenza di bit.

Uno standard: Bus EBU/SMPTE (ES-Bus)

L'intelligenza distribuita, nei device e nelle connessioni, è stata la base per standardizzare controlli remoti del genere e per i piani di studio sia della SMPTE (Society of Motion Pictures and Television Engineering) sia della EBU (European Broadcaster Union), noti organismi internazionali dediti per statuto alla ratifica di standard TV industriali. E normalmente l'industria è fortemente dipendente dalle scelte fatte da tali «authority» al punto di consigliare, spesso con forti pressioni, proprie soluzioni oppure influenzare il mercato con standard «de-facto» non ancora ufficiali, ma ritenuti vincenti. Infatti vi sono sul mercato almeno una dozzina di case costruttrici che usano applicare simili standard in una grande varietà di prodotti audio e video. Ma andiamo prima ad esaminare la terminologia in essere per meglio capire i concetti di fondo.

Supponiamo che ogni unità controllata dal sistema è un device; ogni device dovrà avere una interfaccia intelligente, interna o esterna ad esso, chiamata «tributary» (ausiliare) ed un certo numero di device, controllati attraverso i loro ausiliari, saranno connessi ad un controllo locale o «interface bus» per formare una rete locale (figura 2).

La rete locale sarà gestita da un bus controller, che supervisionerà le comunicazioni tra i device connessi alla stessa. Il bus di interfaccia sarà quindi il canale di comunicazione tra ogni tributary ed il bus controller. Diverse reti locali potranno essere unite insieme mediante un canale con l'uso di un «bus interconnection», che permetterà l'interazione tra i tributary (ogni device) e le diverse reti locali. Ogni rete locale poi includerà un «gateway» (accesso) per tradurre i protocolli della rete locale al protocollo del bus di interconnessione.

Teoria delle comunicazioni

Ogni sistema di comunicazione può essere visto come un insieme di «layers» (piani diversi) che combinati tra loro formano l'intera struttura (figura 3). Ogni layer di grado più alto aggiunge valore alla qualità del servizio reso, rispetto a quelli di grado più basso. Il valore aggiunto sarà stabilito da una «entity» (entità) residente a quel grado di layer. Due entità che operano nello stesso piano, ma in luoghi differenti della rete, saranno chiamati «peer» (pari) e nel designare un sistema si dovrà tener conto maggiormente proprio di questo, cioè permettere comunicazioni tra pari entità.

In modo ideale, il path di comunicazione tra esse sarà di tipo virtuale, che in sintesi significa una connessione «trasparente» che le lascia ignare ai layer di grado più basso.

In realtà, comunque, il path di comunicazione passa attraverso i layer di più basso livello ed anche, ovviamente, nel comune mezzo fisico di connessione.

Come ben evidenziato in figura 4, un sistema di controllo teorico consta di sette layer. Il settimo, detto «Application layer» si occuperà di controllare il device connesso, ed a sua volta sarà controllato dallo stesso device. Da notare che nello standard ES-Bus questo livello non è presente, così come pure sono ristrutturati anche gli altri sei a soli quattro livelli, che andiamo a vedere in dettaglio.

Il «Virtual machine level» o layer di presentazione è l'entità che risponderà a dei dati in una certa maniera, a prescindere dalle caratteristiche specifiche del device collegato. E diversi «dialetti» sul linguaggio possono esistere per ogni tipo di «macchina virtuale».

Il «System service level» convertirà l'indirizzamento logico in quello fisico, ed il dialetto di ogni macchina sarà così identificato. I messaggi saranno in forma assemblata, se la macchina è un device che controlla, oppure segmentata per l'uso immediato, se la macchina è un device controllato.

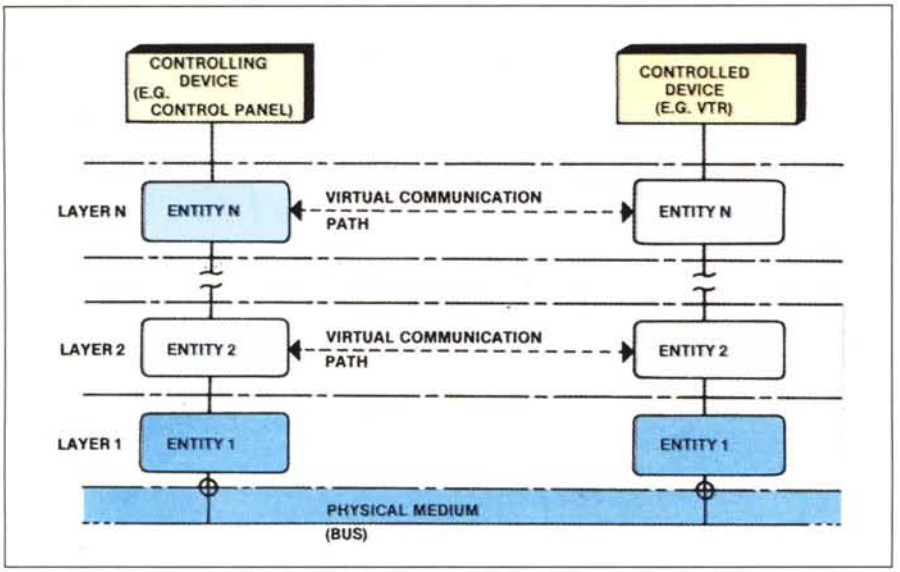


Figura 3 - Una entità, o funzione, di un device potrà dialogare con una uguale entità di un altro device attraverso un path virtuale. In realtà le comunicazioni dovranno passare sia attraverso tutti i layer di grado più basso che nel bus fisico.

Nel «Supervisory level» invece sarà consentita la sincronizzazione dei dati ed il loro invio, mentre la detenzione di errori inizierà una richiesta di ri-trasmissione. L'accesso ai tributary sarà permesso solo dopo che, verificata la scansione di rete (polling) o l'interrogazione del bus controller, essa sia libera da ogni altra attività.

Per finire, il «Physical level» è il «mezzo elettrico» del canale di comunicazione o il vero e proprio bus fisico, con le sue precipue caratteristiche.

Messaggi di controllo

Lo scopo principale, nel realizzare un sistema di controllo remoto standard, sarà di interconnettere device diversi, e di diversa produzione, il più semplicemente possibile. E per realizzare tale scopo dovranno essere standardizzate cose come le caratteristiche di base meccaniche ed elettriche, ed allo stesso tempo tutte le funzioni di controllo dovranno usare un set di istruzioni accettabili per ogni dispositivo collegato.

Sono quindi definiti due gruppi di messaggi; i messaggi «virtual machine», usati per passare comandi e ricevere risposte tra le macchine virtuali, che a loro volta sono ulteriormente suddivisi in «common message» (compresi da tutti i tipi di device), in «type-specific message» (ad uso di particolari categorie di device) e in «user-defined message» (specifiche istruzioni della casa costruttrice dell'apparecchiatura non inclusa nella categoria precedente).

Per ogni device collegato alla rete, la combinazione dei tre sub-gruppi di messaggi fornirà il suo cosiddetto «dialetto», e differenti tipi di essi si trovano in device come VTR, ATR, telecinema, switcher, trasmettitori, ecc. Quello di cui si avrà bisogno, comunque, sarà la necessità di rispondere allo stesso set di comandi inviato sulla rete, indipendentemente dal device interrogato.

Il secondo maggior gruppo di messaggi viene denominato «System service messages» ed esso provvederà all'amministrazione della rete essendo a carico del bus controller della stessa.

Struttura del messaggio

La sua struttura sarà di tipo formato fisso, simile a: ***messaggio = Keyword + argomento***. La keyword ad 1 byte specificherà la funzione che vorremmo realizzare, mentre l'argomento includerà uno o più byte di informazione significativa e può non essere richiesto per tutte le funzioni permesse. Per fare un esempio, lo Stop di un VTR avrà bisogno della sola keyword, mentre funzioni evolute come un comando VTR «Fast Forward» (avanti veloce) necessiterà di una keyword (shuttle), un parametro identificativo (shuttle speed) ed un parametro valutativo (fast). In questo caso, comunque, l'identificativo sarà compreso nella keyword e l'argomento può essere ridotto al suo valutativo, cioè: ***messaggio = shuttle + fast.***

Messaggi di controllo di altri tipi di device – switcher, mixer audio e video, generatori di effetti digitali, ecc. – possono differire nel nome dei parametri e nei qualificanti per differenziarli tra loro.

Finora tutto ciò che è stato detto vale per lo standard ES-Bus EBU/SMPTE; considerando che esso è stato definito intorno alla metà degli anno '80, quando la tecnologia digitale era in pieno sviluppo, e la necessità di una normativa, molto sentita dalla grande utenza broadcast era richiesta perentoriamente, esso è divenuto per diverso tempo il riferimento principale delle ulteriori evoluzioni avvenute nel corso di questi ultimi anni. Evoluzioni che hanno prima «rifondato» e snellito le basi dell'ES-Bus, e che poi hanno intrapreso tutt'altro corso. È il caso dello standard Dynabus e del Louth Automation OOP.

Utah Scientific Dynabus

Qualche tempo dopo la ratifica dello standard ES-Bus ci si rese conto che l'evoluzione dei device, e delle loro intelligenze, era cresciuta di molto come pure la loro velocità di intervento ad azioni congiunte. Ma c'erano anche altri fattori che determinarono l'attenzione delle case costruttrici nel modificare il tutto per una più stabile solidità, e per almeno tre buoni motivi; il primo era l'uso, nell'ES-Bus, di un solo processore centrale che supervisionava l'intera operatività del sistema, coadiuvato sì da un secondo in stand-by che all'occorrenza poteva intervenire, ma che aumentava i costi dell'implementazione di molto, ed un altro nel data rate specifico scelto dallo standard – 38,4 Kbaud – che in sistemi molto potenti con condivisione di risorse e bus multiwire induceva ad una limitazione di operatività per l'inevitabile riduzione di velocità. La terza causa poteva essere identificata nel caso di sconnessione di un device dall'ES-Bus, device che richiede per essere unito in rete di un connettore multibus «loop-through» contenente complessi splitting di rete o circuiti di bypass, pena l'interruzione del servizio.

Alla Utah Scientific americana, nota casa costruttrice di sistemi switching, iniziarono a porsi il problema di analizzarne le soluzioni adatte. Dopo aver testato tutte le possibilità offerte in ambito rete locale, ed aver scartato soluzioni come la token ring, la polling ed altro, giunsero a concepire un sistema ragionevolmente economico, che operava ad alta velocità, affidabile ed immune a problemi di «single-point failure». Tutto ciò sotto l'egida di una tecnologia CSMA/CD (Carrier Sense, Multiple Ac-

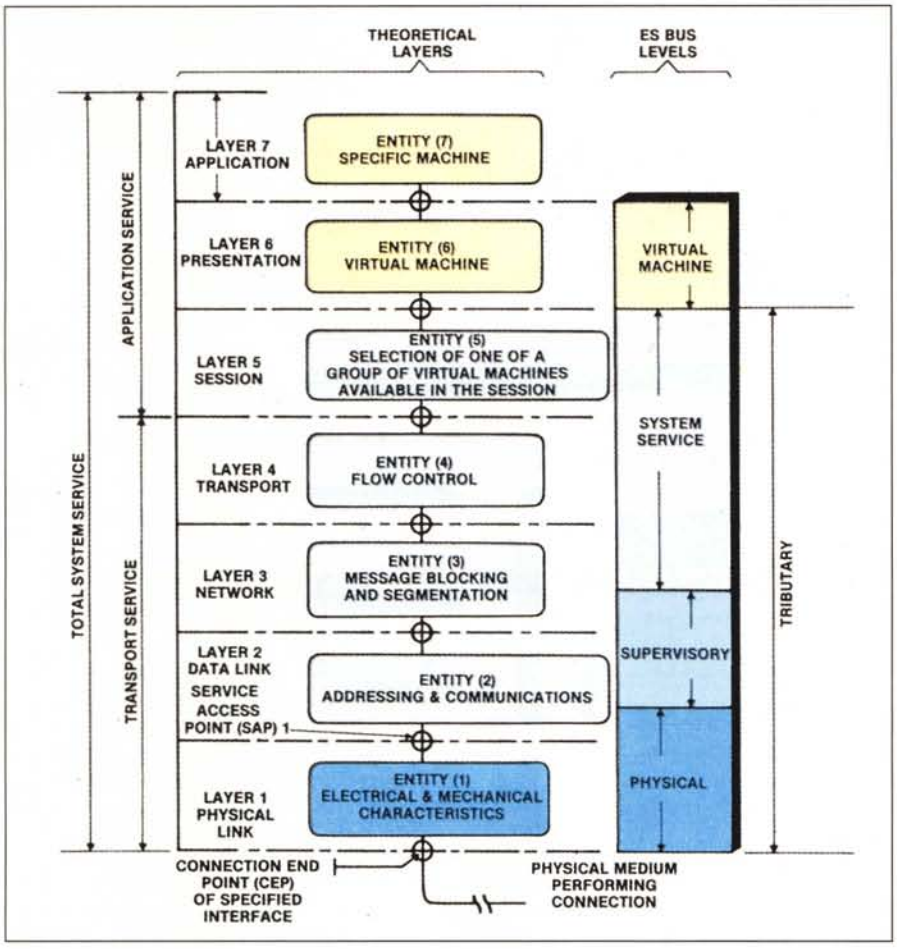


Figura 4 - La teoria delle comunicazioni dividerà il processo di controllo remoto in sette piani (layer) diversi. L'ES-Bus in tale concetto ridividerà i layer in quattro livelli. L'ultimo livello, il settimo, non avrà specifiche in quanto dedicato alle caratteristiche proprie della macchina ed è risolto a livello di «virtual machine».

cess/Collision Detection) e che venne designato commercialmente come Dynabus.

La struttura del sistema non si evolve attorno ad un processore centrale, e nessun singolo device attivo è essenziale al successo dell'operatività della rete. Sono presenti data buffer che isolano le connessioni quando può accadere la rottura di un cavo e dei by-pass che disconnettono il device dal bus se accade di doverlo rimuovere, mentre altri dispositivi rileveranno quando occorre una «collision detection» di dati, invece che controllare l'assenza di un segnale ACK (Acknowledgement) come accade nelle implementazioni tradizionali.

La rete permetterà, previa unità centrale, assegnazioni random di macchine, eliminando l'inefficienza inerente a device dedicati, e sarà amministrata da un semplice terminale di computer che a sua volta opererà in diagnostica nell'analizzare gli status del sistema, rilevare failure e riconfigurare, nel numero e nell'utilizzo di device, aree servite dalla stessa rete per gli scopi più diversi.

La struttura Dynabus

Prima di tutto ci sarà da dire che esso opera attraverso cavi twinax a 150 ohm i cui segmenti non dovranno essere più lunghi di 300 metri. Diversi device controllati potranno essere connessi ad ogni segmento e comunicare con gli altri attraverso questo rudimentale link, mentre gli stessi segmenti potranno essere interconnessi tra loro tramite ripetitori. Due reti diverse saranno unite ad un nodo ed un relativo controller si incaricherà di supervisionare i movimenti dei messaggi da una rete all'altra. Anche se un tipico impiego broadcast probabilmente si limiterà ad una singola rete, la tipologia del sistema consente fino a 226 reti diverse interconnesse, disposte sia in colonna (backbone) che a spina, fino a 15 nodi o link «sub-network» principali. Ogni sub-network poi potrà contenere un massimo di 14 node controller che amministrano le minori sub-network (figura 5); a sua volta un solo segmento potrà sopportare fino a 100 device collegati, mentre ogni rete potrà indirizzare 32.767 device. Per fare un esempio estremo, potremmo dire che la implementazione completa al massimo delle possibilità (15 sub-network principali ognuna con 14 minori sub-network completamente caricate) ci danno il ragguardevole numero massimo di 6.881.085 device controllati, ed indirizzabili, da una singola rete.

Indirizzamento

Qualsiasi device nel sistema Dynabus ha uno o più gruppi di indirizzamento, permettendo ad una macchina di comunicare con un gruppo di device in una sola sequenza; ed ogni device può appartenere ad ognuno dei 32.767 differenti gruppi. Il movimento dei messaggi sarà determinato maggiormente dai nodi presenti, che in questo caso coinciderà con il byte di indirizzo più significativo. Un indirizzo consisterà di 24 bit, come in figura 6, dove avremo i primi 8 bit che formano l'indirizzo di nodo ed i seguenti 16 che identificano l'indirizzo del device locale (LDA). I messaggi poi dovranno contenere entrambi gli indirizzi di designazione (DA) e di sorgente (SA). Per convenzione sono stati assegnati a gruppi differenti di device indirizzamenti diversi. Avremo così che il sistema di controllo delle macchine sarà assegnato ai primi due indirizzi (dispari) in modo esclusivo. Il controllo delle sorgenti sarà il gruppo numero 1 mentre le unità di esecuzione comandi apparterranno al gruppo 3; gli switcher presenti in rete saranno dedicati ai gruppi 5 e 7, ove il primo identifica l'indirizzo di destinazione del messaggio di status ed il secondo i comandi di azione dello switching.

Costruzione del messaggio

La sua struttura consisterà in un preambolo (P), un delimitatore di inizio di frame (SFD), un «field» (campo) DA, uno SA, un «message ID» (MID), un campo di dati (D) ed una sequenza «frame check» (FCS). Quest'ultima è sinonimo di CRC per la correzione degli errori.

Il campo P aiuterà a sincronizzare tutti i device presenti a ricevere i dati trasmessi, e per il clock generale; una durata di tre byte, di solito, determinerà la sua lunghezza fissata dalla più complessa rete presente nel sistema. In questo campo vi sarà una sequenza ripetitiva di 1 e di 0 la quale dovrà iniziare con un 1 e finire con uno 0.

Il pacchetto SFD segnalerà la fine del preambolo P con un univoco pattern di bit (10101011) e tale campo indicherà anche l'inizio del seguente frame di trasmissione.

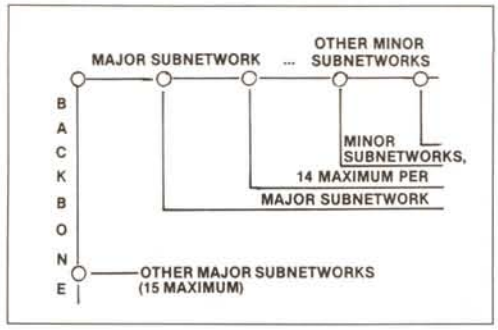
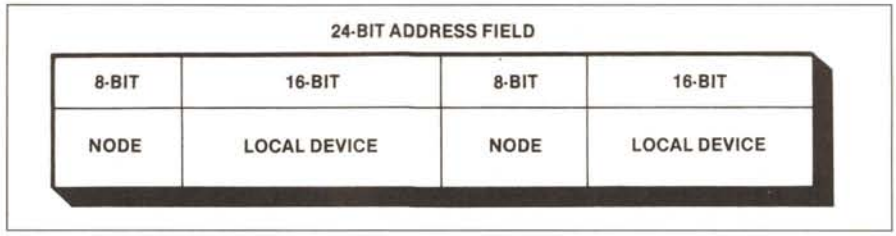


Figura 5 - Nell'implementazione Dynabus i node controller supervisioneranno il passaggio delle informazioni da una parte all'altra della rete.

Figura 6 - La struttura dell'indirizzamento Dynabus, in cui sono contenuti dati sul nodo interrogato e sulle regioni locali della rete che riguardano i device connessi.



Nel campo DA 3 byte provvederanno all'indirizzamento della destinazione del messaggio al device o al gruppo di essi. Sarà a sua volta seguito da un campo SA di altri 3 byte per permettere al device ricevente di determinare la sorgente del messaggio, così come ottenere una veloce comunicazione a 2 vie.

Il field MID definirà il tipo di messaggio come di sistema o specifico DA; il bit più significativo indicherà il tipo, mentre i sette meno significativi rappresenteranno il tipo di selezione del messaggio.

Tutti i device connessi alla rete dovranno ricevere e rispondere a tutti i messaggi di sistema, ed i messaggi diretti a degli specifici DA saranno consentiti solo a quei device. Il formato dei rimanenti sette bit saranno definiti dalle case costruttrici delle apparecchiature.

Il campo D conterrà un «cue» (suggerimento), per il device ricevente, nel dare un senso al messaggio ricevuto. Il formato del campo D sarà definito da quello MID ed esso potrà essere non più lungo di 256 byte.

L'uso del campo FCS sarà di testare l'integrità delle trasmissioni. Il suo contenuto sarà generato dall'apparecchiatura trasmittente e verificato da quella ricevente. Se i valori coincideranno il dato sarà accettato come corretto. I calcoli dei valori FCS saranno dettati dal primo byte del campo DA e termineranno con l'ultimo byte del campo D. Essi saranno determinati come un polinomio standard CCIT/HDCL ed il suo valore è normalmente calcolato dal controller LAN impiegato.

Per concludere, le categorie dei messaggi sono state definite in 10 gruppi ed operano sulla verifica della risposta finale data dal device, al report di errori, a quello di trasmissione, all'identificazione nominativa in ricezione e trasmissione ed ai messaggi di «on-line», che informano il sistema del device pronto a riceverne.

Il design del Dynabus era stato formulato principalmente per un uso specifico in ambito broadcast e sicuramente provvede allo scopo, soprattutto nella velocità di intervento e nell'affidabilità delle comunicazioni tra device diversi, oltre che nella economicità di fondo; anche se non ha molta similitudine con LAN offerte dall'industria dei computer, senz'altro consente di soddisfare i sempre più crescenti realistici bisogni dei broadcaster in ambiti così critici.

Louth Automation OOP

Ma c'è ancora una variante ai discorsi fatti finora; considerarla variante è certamente riduttivo, come sicuramente possono pensare quelli della Louth americana, giovane casa produttrice di sistemi di controllo, poiché in sintesi il loro approccio al problema rivoluziona totalmente tale concetto. Ed anche alla Louth si sono posti molte domande sulle implementazioni più idonee, soprattutto dopo l'esperienza acquisita negli anni sia dagli apporti di tecnologia sia, soprattutto, con gli strumenti propri della programmazione informatica (leggi software più potente degli anni '80 e linguaggi molto evoluti e flessibili).

Come dire che si è data molta più importanza alla parte gestionale, quindi computer e soprattutto linguaggio d'uso, che alla scelta di rete, che in questo caso non hanno quasi più ragion d'essere. A conferma di tutto ciò, ci sarà da dire che in effetti negli anni '80, all'uscita dello standard ES-Bus, i sistemi erano fortemente «hardware dipendenti» e creavano tutta una serie di problemi connessi ad una scelta, forzata, del genere.

La soluzione per gli anni '90 pare sia quella di scegliere la tecnologia dei nuovi strumenti software, in aggiunta alle potenze crescenti dell'hardware, per sviluppare sistemi custom estremamente robusti e flessibili, in contrasto con le tecnologie usate attualmente. Il sistema Louth quindi è identificato come un «software driven system» ove i device collegati potranno essere usati per svolgere una quantità di funzioni, potranno essere create nuove applicazioni e più importante di tutto aggiunte senza problemi nuove produzioni hardware. La tecnologia del software che può supportare tutto questo è identificata come OOP (Object Oriented Programming), qualcosa di cui sicuramente avrete sentito parlare.

Per riassumere brevemente simile soluzione potremo dire che la programmazione OOP, da qualche anno in costante emergere nel panorama dei linguaggi più evoluti, è un approccio modulare allo sviluppo di software che amplifica molto la produttività, riducendone a sua volta i cicli di sviluppo. Può permettere il riuso di software già esistente, in nuove applicazioni, senza riscrivere ulteriormente codici funzionali comuni all'applicazione.

Uno dei più evidenti vantaggi della OOP è l'abilità di «incapsulare» nuovi oggetti (termine tipico della stessa e che identifica funzioni determinanti una azione) in altri oggetti già presenti estendendone la loro funzionalità. In ambito broadcast, per esempio, device come VTR, ATR, switcher ed altro potranno essere «creati», cioè costruiti nelle loro funzioni fondamentali, attraverso oggetti software, controllandone l'omologo fisico. Una volta che un device di tal genere sarà stato creato, la sua interfaccia verso l'applicazione diventerà di fatto standard, cioè universale per il device fisico cui appartiene.

Per fare un esempio, un «software object» VTR potrà essere definito per svolgere funzioni come Play, Record, Stop, Fast Forward, Rewind, Search, e tutte le applicazioni saranno in grado di usare un «vero» VTR con una simile interfaccia software. Non ci si dovrà più preoccupare di quale tipo di VTR sarà impiegato o come lo stesso verrà controllato a livello di device fisico (interfacce RS-XXX oppure connessioni in rete). L'applicazione è totalmente isolata dai protocolli del device e dalle sue connessioni fisiche. In questo modo è facile notare che se un differente device sarà aggiunto all'applicazione la stessa non dovrà in nessun caso essere variata; la

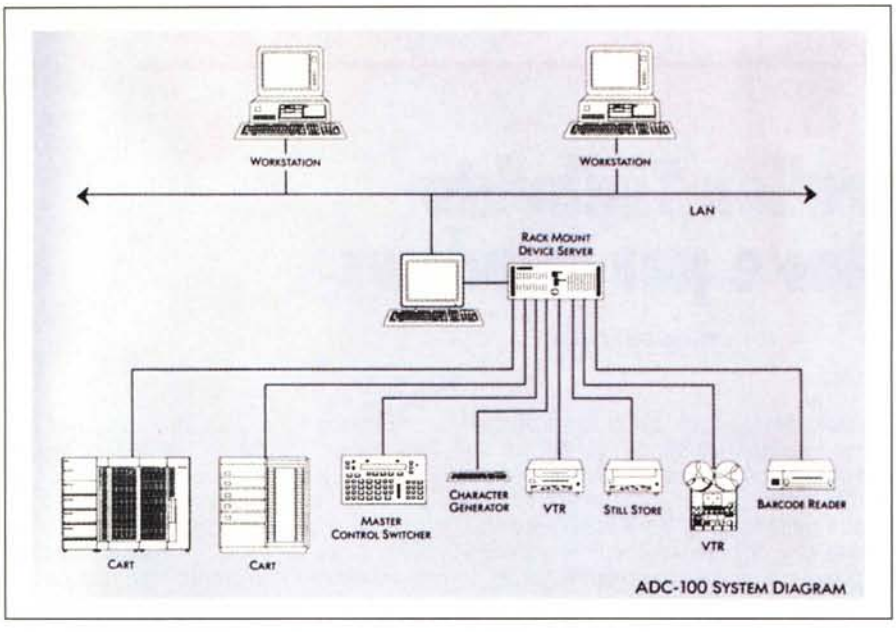


Figura 7 - Tipica applicazione OOP nella tecnologia Louth; è presente una rete, con terminali collegati, un terminale di gestione singolo ed interfacce standard (RS-422) verso i device connessi. Più il software di controllo Object Oriented.

sola ovvia richiesta sarà quella di provvedere a creare un nuovo oggetto software per il nuovo arrivato.

Essi potranno essere modificati indipendentemente dal contesto senza variare il programma di gestione sistema, per un debug efficiente oppure come upgrade dell'applicazione, mentre la stessa potrà essere a sua volta testata e verificata usando oggetti che ne fanno parte senza che i corrispettivi fisici siano presenti.

La tecnologia OOP elimina poi la necessità di interfacce hardware dedicate poiché tutte le funzioni del device saranno replicate nel software; il sistema richiederà solo la presenza di una rete (qualsiasi NetBios LAN) per il controllo totale delle funzioni. E questo in effetti è già molto, considerando l'eliminazione dei prodotti interfacce (scomode e costose) e la necessità, finora, di ricorrere a dei bus esterni (come accade nell'ES-Bus o nel Dynabus) che abbisognano di critiche messe a punto e di protocolli dedicati.

Flessibilità

Dopo tutto quello che è già stato evidenziato, potremo ancora dire che il fatto di inglobare un device come un VTR in un «software object» permetterà a tutte le applicazioni di vedere lo stesso in maniera identica. Nessuna speciale programmazione, da applicazione ad applicazione, sarà necessaria per usarlo. Esso potrà essere assegnato per svolgere varie funzioni in contesti diversi; per esempio nel configurarlo come macchina «Record» ed in seguito se occorre dinamicamente riassegnarlo ad una suite di editing come device «Source» o macchina «Play».

La versatilità a questo punto diventa innegabile.

Introducendo poi il concetto di architettura Client/Server, per applicazioni multiple concorrenti, ancora nessun problema; la rete a cui appartengono i device è a sua volta un oggetto e qualsiasi tipo di LAN, come Novell NetWare, Token Ring o ArtSoft LANtastic sarà in grado di colloquiare con il tutto.

Tali implementazioni useranno procedure remote che interrogano il sistema a livello di rete e questa tecnologia ci consentirà di usare interfacce tipiche API (Application Programming Interface) indipendenti dal linguaggio usato.

Ridondanza

La capacità di sopperire a guasti di device nel sistema sarà facilmente praticabile poiché lo stesso verrà sostituito inserendone un altro. Nel caso di un VTR esso potrà anche essere di tipo o fattura diversa, proveniente da altri contesti in cui svolgeva altre funzioni; la dinamicità della re-assignazione di device è fatta salva poiché per l'applicazione in corso, e per la tecnologia OOP, un VTR è sempre e solo un VTR. Tutto ciò tranquillamente anche in casi Client/Server; se un client rileva un guasto nel server potrà diventare se stesso un server e automaticamente impadronirsi delle linee di controllo del server guasto a suo beneficio. A causa rimossa tutto potrà tornare come prima.

A prova di futuro

Lo sviluppo di tecniche di programmazione OOP è senz'altro la chiave vincente dove è necessario controllare device diversi o reti connesse tra loro nello svolgere funzioni concorrenti. Nella fattispecie, e parlo del settore broadcast, ove le innate idiosincrasie verso nuove tecnologie limitano molto le decisioni finali a causa dei costi sempre più pesanti, la tecnologia OOP può veramente rendere il tutto «a prova di futuro». Quando nuovi device saranno prodotti o nuove applicazioni potranno essere prese in considerazione, il sistema sarà facilmente espanso a tali bisogni senza la necessità di dover mettere mano ad architetture già funzionali.

Conclusioni

Sperando che anche questa seconda parte non sia stata tediosa e astrusa nei concetti e nella forma (e vi assicuro che la materia è molto più complessa di quanto evidenziato in queste note) si potrebbero tirare delle conclusioni.

I sistemi di controllo remoto broadcast sono strumenti ormai da tempo indispensabili alla gestione di risorse nell'economia di medio-grandi utenze TV. Pensando solo che comprendono dalla semplice suite di editing, costretta in una stanza, a sistemi che occupano interi edifici in aree ben distinte tutte dialoganti tra loro (come editing, processione audio-video, aree news ove occorre accesso a librerie di materiale video girato, collegamenti via satellite e trasmissioni singole o in rete) la necessità di un sistema centrale è ovvia.

Ed i problemi sono ancora molti e non tutti affrontati esaustivamente. L'uso delle reti locali è già una realtà consolidata, e non potrebbe essere diverso, ma l'approccio al governo dei device, o in altri casi di intere aree, non è ancora stabilmente verificata, anche se l'estensione delle nuove tecnologie software potrà fare molto per dirimere adeguatamente la questione. La programmazione OOP di sicuro è uno strumento molto efficace allo scopo, ma attende quantomeno una standardizzazione che forse sarà difficile da ottenere. C'è da non dimenticare che gli organismi preposti allo studio di tali cose sono fortemente «dipendenti», se si può usare un simile termine, dalle proposte dell'industria del settore. Ed affidare al «solo» software l'indirizzo di uno standard è di sicuro una proposta non praticabile. Non rimane che attendere qualche tempo per vedere consolidata questa nostra prospettiva. MC