Corso di Calcolatori Elettronici II

ASIM

Ambiente per la SImulazione dei sistemi a Microprocessore

Prof. Antonino Mazzeo

Parte I-descrizione funzionale e modalità d'uso di ASIM

1 CARATTERISTICHE PRINCIPALI DI ASIM

ASIM (**A**mbiente **SI**mulazione **M**icroprocessori) è un ambiente didattico per la simulazione di sistemi a microprocessore e che consente all'utente di definire proprie configurazioni di architetture hw e di estendere l'insieme originario di dispositivi presenti con nuovi componenti sviluppati in ambiente C++.

ASIM è stato sviluppato e si è evoluto a partire dal 1984 in seno ai corsi di Macchine per l'Elaborazione delle Informazioni e Calcolatori elettronici II, con la collaborazione degli studenti e con il lavoro prezioso di numerosi tesisti di cui voglio ricordare soltanto l'ing. L. Impagliazzo che ha dato un notevole contributo nel portare, ridisegnandolo strutturalmente, il simulatore nell'ambiente C++/Windows. La versione iniziale è stata scritta in C e operava in ambiente MSDOS, è in atto il porting in JAVA nelle piattaforme Linux e Windows.

ASIM è un ambiente di simulazione in cui i sistemi hw da simulare sono caratterizzati da configurazioni di architetture composte dall'interconnessione di oggetti (CPU, memorie, bus, device e nodi) e su cui poter operare sia in modalità mono che multiprocessore con programmi scritti in linguaggio assembler.

Poiché ASIM consente di specificare architetture di vario tipo (mono, multi processore, data flow, shared memory, message passing, etc.) è possibile utilizzare l'ambiente oltre che come un laboratorio per lo studio di sistemi a microprocessore, con l'impiego di programmazione in assembler, anche come un laboratorio per lo studio dei livelli architetturali superiori di sistemi (S.O., multiprogrammazione di basso livello, ecc.). Ad esempio i processori possono essere connessi con bus locali, globali e crossbar, oppure possono formare, insieme ad una memoria, dei nodi di elaborazione; questi possono poi essere connessi tra loro con link per formare strutture ad ipercubo o mesh.

Il limite alla complessità dei sistemi simulabili, in termini di componenti impiegati in una configurazione, è imposto dalla configurazione (capacità di memoria, velocità del processore in uso, ecc.).

Il simulatore ASIM è scitto in C++, opera in ambiente Microsoft Windows, di cui riflette l'organizzazione a finestre e utilizza pienamente le classi offerte da MFC (Microsoft Foundation Classes) per l'interazione con Windows stesso. Esso è organizzato come applicazione di tipo MDI (Multiple Document Interface) cioè un'applicazione con documenti e viste multiple. L'applicazione MFC è caratterizzata da quattro classi principali:

- CasimApp ← CwinApp
- CasimDoc ← Cdocument
- CasimFrame ← CMDIChildWnd
- CasimView ← Cview

A destra della freccia sono indicate le classi di MFC da cui sono ereditate le quattro classi di ASIM. Per i dettagli sull'architettura di ASIM di consulti il documento "ASIM Internal Design".

L'ambiente IDE (Integrated Development Environment) è caratterizzato da una finestra principale (Fig. 1) che delimita l'area di lavoro e presenta un menù da cui è possibile accedere a tutti i comandi. La barra di fondo (barra di stato) indica il numero di eventi (clock) schedulati nella simulazione e il punto di "break", relativamente al numero di clock da eseguire, cioè l'evento che determina l'arresto momentaneo della simulazione. In ASIM, infatti, la simulazione viene scandita dai segnali di "clock" o "passi" inviati dallo schedulatore a tutti gli elementi; l'intervallo minimo di simulazione è, quindi, il clock; il numero di azioni che ogni dispositivo compie in un colpo di clock dipende dalle caratteristiche del dispositivo stesso.

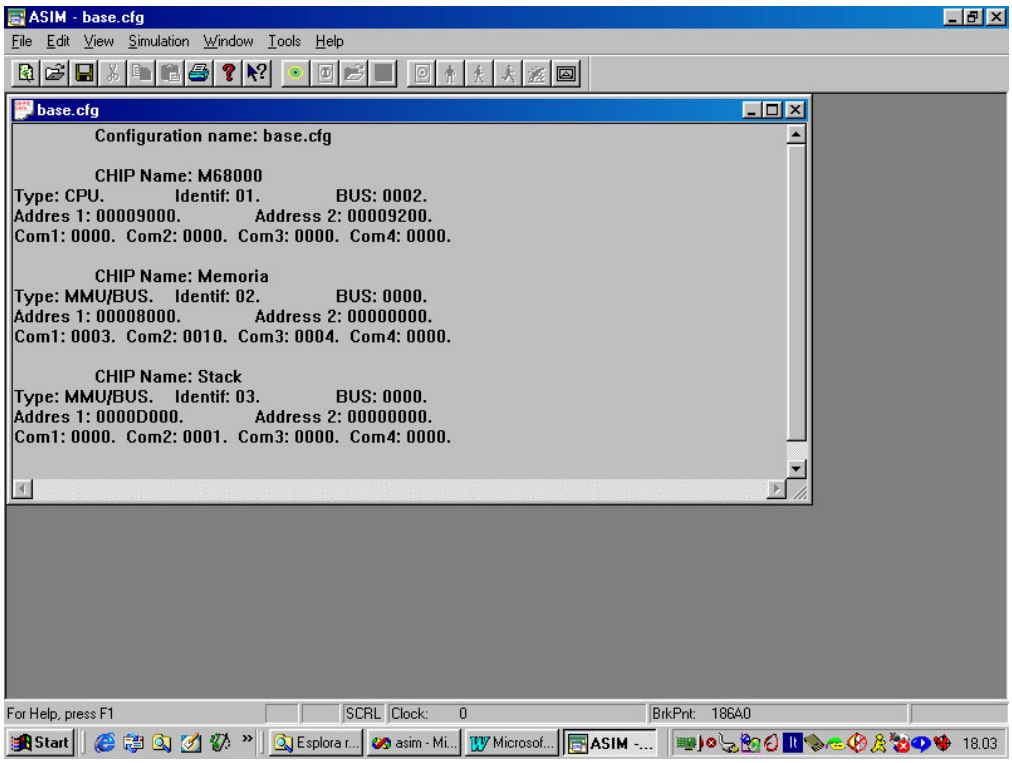


Fig. 1-La finestra principale di ASIM

Ad ogni oggetto componente una configurazione da simulare è associata una finestra in cui vengono presentati i dati che lo caratterizzano coincidenti, in linea di massima, con il modello di programmazione del componente generico (ad es. il contenuto dei vari byte per una memoria o dei registri per un processore). Quando una finestra è attiva (selezionata), il menù sulla barra di controllo viene specializzato per le specifiche funzioni dell'oggetto associato. La finestra corrente può essere ridimensionata, spostata in ogni punto dell'area di lavoro o ridotta a icona (quest'ultima possibilità va sempre sfruttata nel corso di una simulazione, quando non si è interessati alla visualizzazione dei dati di un particolare elemento, in modo da aumentare la velocità della simulazione stessa).

In ASIM tutti i numeri sono rappresentati in notazione esadecimale, uniche eccezioni sono le rappresentazioni in binario per quei registri in cui assumono significato singoli o aggregati di bit (ad esempio il registro di stato di un processore).

Dalla versione 3.0 ASIM offre:

- la gestione di configurazioni multiple;
- l'accesso rapido alle configurazioni usate più di recente;
- il lancio del programma con la configurazione desiderata già aperta chiamando il relativo file *.cfg dal "FileManager" di Windows;

- la creazione di una configurazione mediante il meccanismo di "taglia e incolla" tipico di Windows;
- la visualizzazione completa dei dati relativi a ciascuna configurazione;
- la stampa di detti dati con font di caratteri, intestazione e piè di pagina selezionabili; numerazione automatica delle pagine;
- l'anteprima di stampa delle pagine da inviare alla stampante;
- l'esecuzione veloce di una simulazione;
- la simulazione contemporanea di due configurazioni;
- la simulazione in background mentre si lavora su un'altra configurazione;
- l'accesso rapido ai comandi mediante ToolBar;
- l'help in linea su barra di stato;
- l'help da indice o dipendente dal contesto;
- l'accesso alla "Guida alla Guida" in Windows;
- la protezione da errori d'uso mediante il meccanismo di mascheramento dei comandi.

2 Il meccanismo di simulazione e l'architettura ad oggetti di ASIM

2.1 Il meccanismo di simulazione

Una simulazione in ASIM è caratterizzata da una specifica configurazione di sistema a microprocessori, contenente processori e dispositivi (oggetti) messi a disposizione dall'ambiente. Tutti gli oggetti inclusi in una configurazione vengono inseriti in una lista che il simulatore consulta per sviluppare i singoli passi di simulazione e per effettuare controlli di consistenza delle interfacce dei vari oggetti e i vincoli di interconnessione. Definita la configurazione della macchina da simulare (mediante l'ausilio dell'editor di configurazione) si provvede a "costruirla" premendo il bottone di "costruzione" e ad attivarla premendo il bottone di "accensione" (o l'equivalente voce ON/Reset del Menù Simulazione) della macchina realizzata (Fig. 2).

All'attivazione di una configurazione, sono eseguiti controlli sulle connessioni predisposte tra i dispositivi, in particolare, la finestra principale invia un messaggio (messaggio bloccante Windows WM_RESET) a ciascun dispositivo interessato e i cui dati caratterizzanti sono raccolti in un oggetto della lista e aspetta che questo esegua i necessari controlli. Se viene riscontrato nel dispositivo qualche errore di configurazione, una finestra di dialogo segnala all'utente l'anomalia e viene restituito un codice di errore che interrompe ulteriori controlli e sospende la simulazione. In assenza di errori, la finestra principale invia lo stesso messaggio ad un altro *dispositivo* e, solo se tutti i controlli per ogni componente danno esito positivo, abilita i comandi per la simulazione (per ulteriori dettagli si analizzino i sorgenti dei moduli CMainWindow e CBaseWnd).

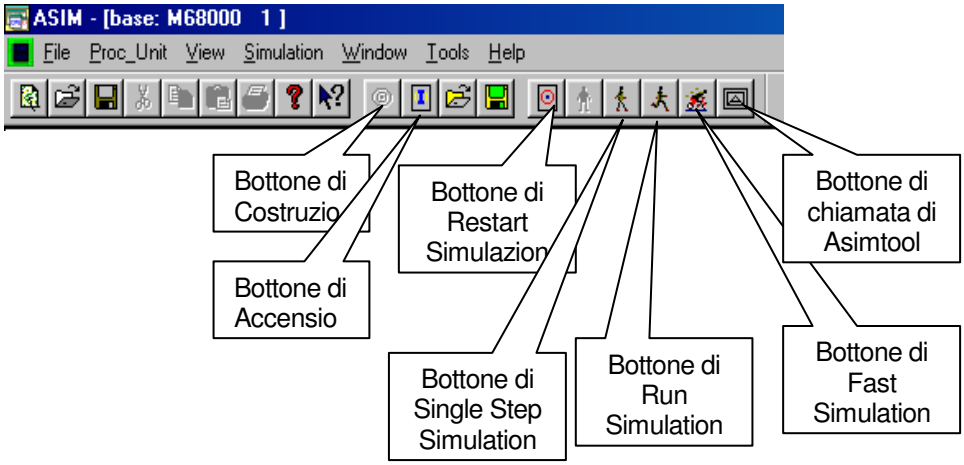


Fig. 2-La barra di controllo di ASIM

In modo analogo è eseguito un passo di simulazione ma con la differenza che i messaggi sono inviati in successione e in modo non bloccante.

Per controllare l'esecuzione di un passo di simulazione la finestra principale utilizza due parametri ClockHigh (TRUE se un passo di simulazione è in corso) e ChildBusy (che indica in ogni istante il numero di dispositivi che stanno eseguendo un passo di simulazione).

Alla richiesta di un passo di simulazione ClockHigh è posto a TRUE e la finestra principale invia a tutti i dispositivi il messaggio WM_CLOCK senza attendere risposta (messaggio non bloccante); ChildBusy assume un valore pari al numero di componenti che formano la configurazione, ad indicare il fatto che tutti i dispositivi stanno IN PARALLELO eseguendo le operazioni previste dal passo di simulazione (operazioni che comportano in generale un'interazione tra i vari dispositivi).

Quando un dispositivo ha completato l'esecuzione del passo di simulazione, un messaggio (WM_COMPL) segnala alla finestra principale (che decrementa ChildBusy) l'evento. Quando tutti i dispositivi hanno completato (ChildBusy è tornato a zero e ClockHigh è riportato a FALSE) un nuovo passo di simulazione può essere richiesto (per maggiori dettagli si consultino i sorgenti del modulo CMainWnd::TraceProg che riporta il codice richiesto per il lancio di un passo di simulazione).

ASIM consente, oltre all'esecuzione di singoli passi di simulazione (Single Step Simulation) su richiesta dell'utente, anche l'esecuzione dei programmi (Run Simulation) in modalità "trace" (come successione di singoli passi) e l'esecuzione completa (Fast Simulation) con cui eseguire velocemente la simulazione senza la visualizzazione dei dati di "trace" .

2.2 Classi di oggetti attualmente simulabili in ASIM

Un sistema calcolatore è costituito da un insieme di componenti (processori, memorie, device, ecc.) fra loro interconnessi. In Asim sono stati implementati un ridotto numero di modelli di componenti da cui poter derivare, secondo il meccanismo di ereditarietà tipica delle classi C++, specifici componenti. Questi, a seconda delle funzionalità svolte e delle modalità di interfacciamento con altri dispositivi, sono stati raggruppati in quattro classi principali. L'appartenenza di un dispositivo ad una di tali classi è subordinata al possesso di almeno una delle proprietà che la caratterizza (nel ricavare le proprietà si è fatto un confronto tra il modello proposto e le reali architetture dei sistemi esistenti). Sono descritte, di seguito, le classi allo stato supportate dal simulatore.

2.3 Classe dei dispositivi Processore

Processore: un qualsiasi dispositivo che ha lo stato interno rappresentato dal contenuto dei suoi registri, più un insieme di informazioni riguardanti il/i bus cui è connesso e le eventuali richieste di interruzioni pervenute e che gode delle seguenti proprietà:

- P1) può eseguire un programma contenuto in una memoria (interna o esterna al dispositivo stesso);
- P2) può accedere attraverso il/i bus cui è connesso ad una memoria (esterna) o ai registri di altri dispositivi per eseguire operazioni di lettura o scrittura;
- P3) può servire e/o gestire interruzioni provenienti da altri dispositivi, eventualmente assegnando ad esse un livello di priorità.

Oltre alle CPU, che sicuramente sono classificabili come processore in quanto godono di tutte le proprietà suddette, rientrano in tale insieme anche i processori video e di IO ed i coprocessori matematici (godono delle proprietà P1 e P2), i DMA (godono della proprietà P2), i "priority interrupt controller" (godono della proprietà P3) ed infine strutture "hardware" per la gestione prioritaria delle interruzioni come il "daisy chaining" (godono della proprietà P3). I processori supportati da ASIM appartengono a tale insieme.

2.4 Classe dei dispositivi Nodo

Nodo: è un dispositivo il cui stato interno raccoglie informazioni relative alle connessioni con altri nodi ed ai messaggi in transito da/verso questi e che gode delle seguenti proprietà:

N1) può consentire la connessione tra dispositivi dello stesso tipo;

N2) può gestire, come nodo intermedio, la connessione tra dispositivi dello stesso tipo, instradando la comunicazione;

N3) può avere capacità autonome di elaborazione e trasformare (secondo una logica qualsiasi) i messaggi o dati ricevuti.

N4) può essere connesso ad un bus/memoria.

2.5 Classe dei dispositivi Bus/Memoria

Bus/memoria: un dispositivo il cui stato interno è caratterizzato dai valori assunti da un insieme molto ampio di registri e/o da informazioni riguardanti i dispositivi che esso connette e che gode delle seguenti proprietà:

M1) può consentire a dispositivi di tipo processore di leggere o scrivere i suoi registri interni (la selezione avviene in base all'indirizzo);

M2) può consentire a dispositivi di tipo processore di leggere o scrivere i registri dei dispositivi di tipo device (la selezione avviene in base all'indirizzo);

M3) regola l'accesso di più dispositivi di tipo processore ai suoi registri ed ai device (in ogni istante al più un accesso è in corso, gli altri processori devono attendere);

M4) gestisce la memoria fisica effettuando il “mapping” degli indirizzi virtuali negli indirizzi fisici. M5) può consentire l'esecuzione di cicli non interrompibili di lettura-modifica;

M6) può connettersi ad altri dispositivi dello stesso tipo sia per costruire accessi da un altro bus sia verso un altro bus;

2.6 Classe dei dispositivi device

Device: tutti i dispositivi che non rientrano nelle tre precedenti classi e che godono delle seguenti proprietà:

D1) può essere connesso ad un bus/memoria in modo che un processore possa accedere ad esso;

D2) può essere in grado di generare delle interruzioni da inviare ad un processore; nel qual caso deve essere connesso in qualche modo al processore;

D3) può essere connesso e comunicare con un altro device della stessa macchina o di un'altra macchina;

D4) può connettere dispositivi di tipo processore e bus/memoria a dispositivi del tipo bus/memoria.

3 Manuale d'uso di ASIM

3.1 I Menù della barra di Controllo di ASIM

3.1.1 Il menù "File".

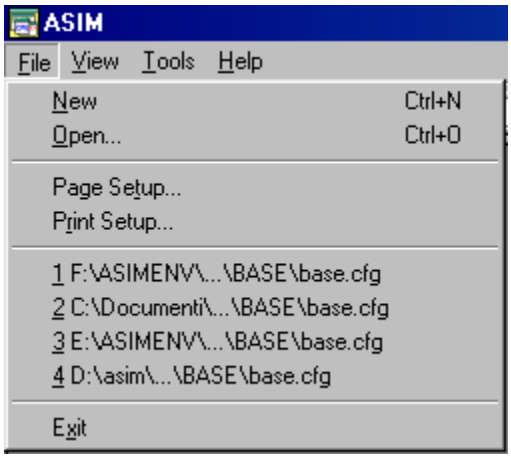


Fig. 3-Il Menù File

Dal menù **File** (Fig. 3) è possibile accedere alle tradizionali funzioni di apertura (nuova o già esistente) di una configurazione, alla stampa della stessa e alla chiusura di ASIM.

New consente di creare una nuova configurazione aprendo una finestra ed attivando per essa il menù Edit (paragrafo 1.2); da questo menù sarà possibile selezionare tutti i comandi necessari a definire i componenti della nuova configurazione. Poiché è possibile avere più configurazioni aperte contemporaneamente, il comando New non chiude alcuna precedente configurazione.

Open apre un file di configurazione precedentemente salvato; non chiude alcuna precedente configurazione.

Close chiude la configurazione corrente; se vi sono state modifiche chiede se il file deve essere aggiornato.

Save, Save As salva il file di configurazione corrente; Save As consente di specificare un nuovo nome per il file che diventa anche il nome della configurazione corrente.

Print attivo solo se la finestra corrente è una di quelle che mostra i dati della configurazione; il comando consente di stampare questi dati con un set di caratteri selezionabile. Se è necessario utilizzare più di una pagina per stampare i dati, la paginazione è automatica.

Page Setup consente di impostare l'intestazione ed il piè di pagina; tipicamente questi riportano rispettivamente il nome del file di configurazione ed il numero di pagina.

Print Preview questo comando fornisce un'anteprima di stampa ovvero consente di vedere l'effetto della stampa della configurazione corrente sulla stampante e con le impostazioni selezionate.

Print Setup imposta la stampante.

1 .. 4 (Most Recent Used) nome dei file di configurazione usati più di recente; consente di aprire le relative configurazioni immediatamente, senza passare per il comando Open.

Exit, esce da ASIM; per tutte le configurazioni modificate viene richiesto se salvare o meno le modifiche.

3.1.2 Il menù "Edit".

Dal menù **Edit** è possibile accedere a tutti i comandi che consentono di definire una configurazione di sistema; tale configurazione rappresenta la macchina da simulare. Per convenzione i file che contengono le informazioni relative ad una configurazione hanno come estensione ".cfg".

Il menù si presenta come in Fig. 4.

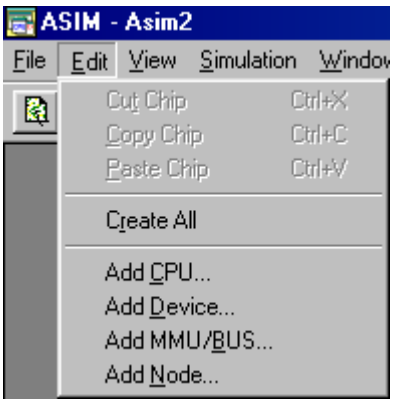


Fig. 4-Il Menù Edit

Questo menù è disponibile solo se la finestra corrente è quella che mostra i dati relativi alla configurazione (corrente); esso mette a disposizione tutti i comandi utili a costruire la macchina da simulare.

Cut Chip elimina il componente selezionato dalla configurazione corrente e salva i dati ad esso relativi nella Clipboard; con il comando Paste Chip sarà possibile inserire questo componente in un'altra configurazione. La selezione del componente avviene con un doppio click del mouse nell'area della finestra che contiene i dati del chip desiderato.

Copy Chip vale quanto detto per Cut Chip con l'unica differenza che il componente selezionato non viene eliminato dalla configurazione corrente ma se ne crea una copia.

Paste Chip consente di inserire in una configurazione un componente i cui dati sono disponibili nella Clipboard; grazie a questo comando ed ai due precedenti è possibile creare nuove configurazioni a partire da precedenti in maniera molto semplice attraverso il tipico meccanismo "copia e incolla" di Windows.

Create All dalla versione 2.0 di ASIM in poi non necessariamente viene creata insieme al componente anche la finestra ad esso associata; ciò per evitare di affollare il video di finestre non utili. È possibile quindi costruire prima una macchina con i comandi di "copia e incolla" o di "Add ..." e poi creare tutte le finestre associate ai componenti con questo comando. È indispensabile eseguire Create All per poter rendere disponibili i comandi relativi alla simulazione.

Add CPU, Add Device, Add MMU/BUS, Add Node consentono di aggiungere alla configurazione corrente un nuovo componente. La finestra associata al nuovo componente è creata immediatamente solo se è stato dato in precedenza il comando Create All per la configurazione corrente.

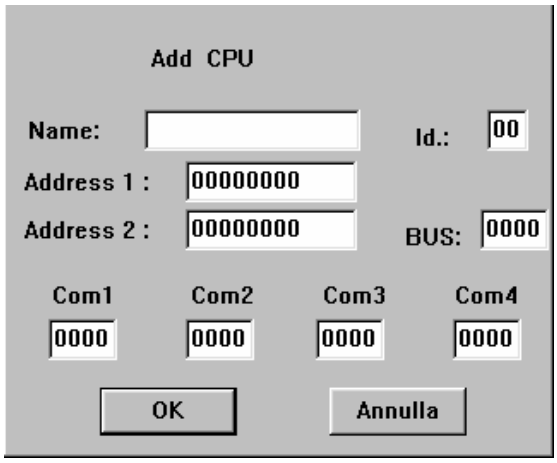


Fig. 5-Menù di Configurazione Oggetti

I comandi Add_CPU, Add_Device, Add_MMU/BUS, Add_Node, danno accesso ad una finestra di dialogo () dove possono essere specificati il nome, l' identificatore ed una serie di parametri che caratterizzano l'elemento da inserire nella configurazione corrente. L' identificatore deve essere un numero compreso tra $01 e $FF e ciascun dispositivo deve avere un identificatore diverso da ogni altro (ciò limita a 255 i componenti appartenenti ad una configurazione); ogni qual volta, nel definire le connessioni tra i vari dispositivi, è necessario riferirsi ad un certo elemento, ciò viene fatto specificandone l'identificatore. Il significato degli altri parametri è dipendente dal particolare componente e sarà chiarito di volta in volta.

3.1.3 Il menù **View (*)**.

Il menù View (Fig. 6) abilita/disabilita alcune funzionalità offerte dalle classi MFC quali toolbar, barre di stato, help in linea, selezione dei font di caratteri.

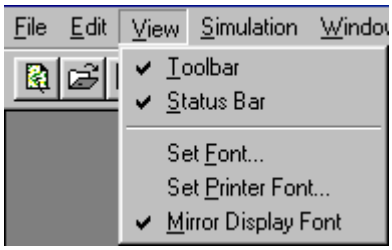


Fig. 6-Il Menù View

Toolbar (*), mostra/nasconde la barra che consente l'accesso rapido ad alcuni comandi (toolbar); i tasti presenti nel toolbar consentono (nell'ordine da sinistra a destra) di attivare immediatamente i comandi: New, Open, Save, Cut, Copy, Paste, Print, Index, Help, Restart, Stop, Trace, Run, Fast Run, ASIM Tool.

Status Bar(*), mostra/nasconde la barra di stato; quest'ultima mette a disposizione dell'utente un "help in linea" ovvero presenta a video una frase che indica l'effetto dell'azione che si sta compiendo o si sta per compiere (selezione di un comando, spostamento di una finestra, ecc.). La barra di stato mostra inoltre due informazioni relative alla configurazione corrente: numero dell'ultimo passo di simulazione eseguito (clock), numero dell'ultimo passo di simulazione da eseguire prima di interrompere la simulazione (BreakPoint); queste informazioni sono visibili solo se sono attivi, rispettivamente, i tasti "Caps Lock" e "Num Lock".

Set Font (*), consente di selezionare il font di caratteri utilizzato dalla finestra che mostra i dati relativi alla configurazione; quest'ultima deve essere la finestra corrente affinché il comando sia disponibile.

Set Printer Font (*), consente di selezionare il font di caratteri utilizzato dalla stampante; affinché il comando sia disponibile la finestra corrente deve essere quella contenente i dati relativi alla configurazione. È ammesso che il font selezionato per la stampante sia diverso da quello selezionato per lo schermo.

Mirror Display Font (*), forza la stampante ad utilizzare lo stesso font di caratteri selezionato per il video con il comando Set Font.

3.1.4 Il menù **Simulation**.

Il menù Simulation () rende accessibili i comandi per il controllo della simulazione di configurazioni.

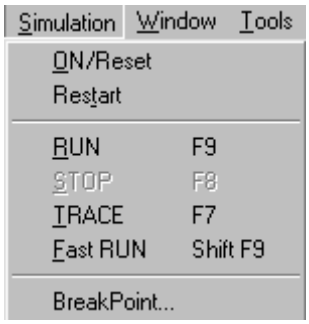


Fig. 7-Il Menù Simulazione

ON/Reset, accende/resetta la macchina da simulare, controlla le connessioni. ON/Reset è abilitato solo se, per la configurazione corrente, è stato dato il comando Create All del menù Edit (paragrafo 1.2).

Restart riporta la macchina nelle condizioni in cui si trovava prima di avviare la simulazione.

RUN lancia una simulazione. ASIM consente anche l'esecuzione di due simulazioni in parallelo oppure di lavorare con una configurazione mentre è in corso la simulazione per un'altra configurazione.

STOP, arresta la simulazione in corso.

TRACE, lancia un singolo passo di simulazione.

Fast RUN esegue le azioni previste dal comando RUN ma prima riduce ad icona tutte le finestre associate ai componenti presenti nella configurazione corrente; ciò consente di accelerare notevolmente l'esecuzione di una simulazione.

BreakPoint, fissa il numero di passi dopo cui bloccare automaticamente la simulazione.

3.1.5 Il menù **Window**.

Questo menù (Fig. 8) raccoglie i comandi utili alla gestione dell'ambiente a finestre multiple.

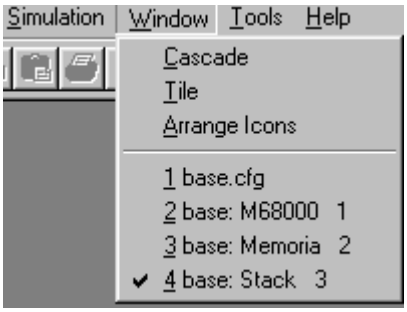


Fig. 8-Il Menù Window

Cascade, dispone le finestre in cascata.

Tile, dispone le finestre affiancate.

Arrange Icons, risistema le icone sul fondo della finestra principale di ASIM.

1..n, seleziona una finestra portandola in primo piano (finestra corrente).

3.1.6 Il menù **Help**.

Questo menù (Fig. 9) raccoglie i comandi utili ad ottenere informazioni su ASIM e sul suo utilizzo.

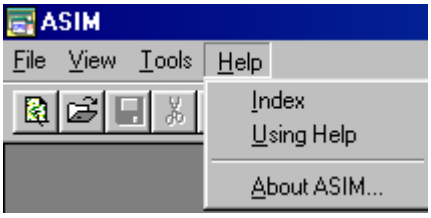


Fig. 9-Il Menù Help

Index (*), consente di accedere alla guida all'uso di ASIM. Attualmente ASIM 2.0 offre tutto il supporto necessario a livello di codice per realizzare sia una guida all'uso sia un help dipendente dal contesto; resta da compilare il file di testo contenente le informazioni.

Using Help (*), consente di accedere a tutte le informazioni necessarie per l'uso di una guida in ambiente Windows.

About ASIM, mostra le informazioni relative ad ASIM (autore, versione, ecc.).

3.2 La barra degli strumenti di ASIM

La barra degli strumenti di Asim (Fig. 10) presenta 19 bottoni raggruppati in 3 gruppi. I 9 bottoni del primo gruppo servono a selezionare le tradizionali funzioni di Windows. I 4 bottoni del secondo gruppo servono ad attivare la configurazione e a ripristinare e salvare la configurazione di lavoro (workspace), i primi 5 bottoni del terzo gruppo servono a selezionare le funzioni di Asim di stop, arresto, esecuzione di un passo, esecuzione ed esecuzione veloce iconizzata di una simulazione. L'ultimo bottone serve ad attivare l'ambiente assembler detto Asimtool.

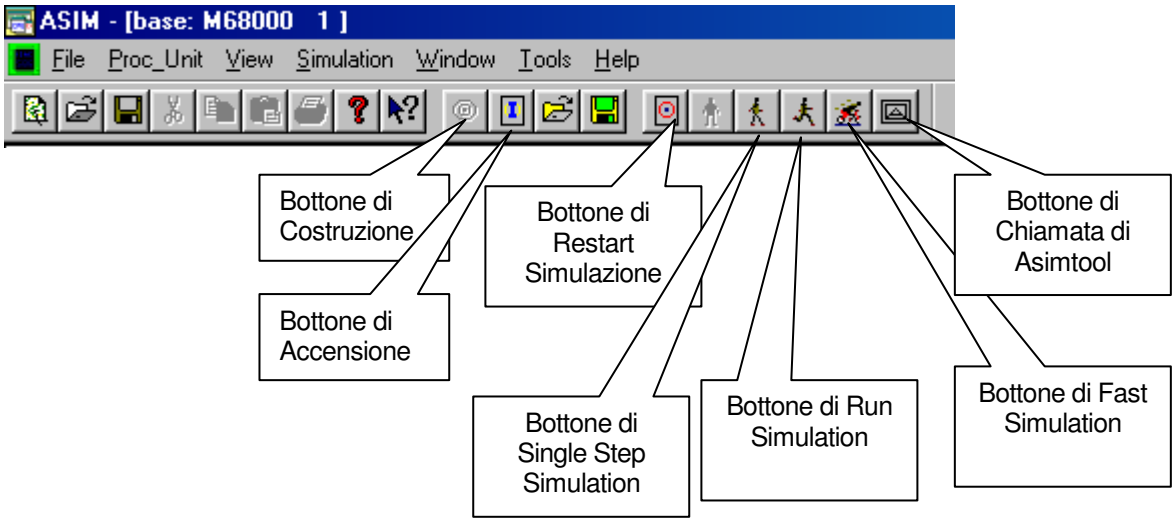


Fig. 10-La barra degli strumenti di ASIM

Taluni dei comandi indicati nei bottoni e nei Menù di Asim possono essere attivati ricorrendo all'uso dei tasti funzione così come riportato negli stessi menù. I comandi possono, infine, essere attivati premendo il tasto ALT e il carattere sottolineato riportato nel nome dei comandi nei vari Menù.

4 CONFIGURAZIONE E SIMULAZIONE DI UN SISTEMA E DEGLI OGGETTI DI ASIM

In Asim un sistema è descritto mediante una configurazione contenente un insieme di oggetti appartenenti ad una delle classi sopra descritte. L'editor di configurazioni di ASIM, per tramite di una "finestra di dialogo", consente di immettere tutti i parametri necessari a descrivere i singoli oggetti ed il loro interfacciamento. La finestra di dialogo presenta una maschera comune a tutti gli oggetti. Tale maschera è dotata di 9 campi che, a seconda dell'oggetto, codificano informazioni atte a caratterizzarlo all'interno della configurazione hw e, quindi, della simulazione. Il significato di tali campi è, in generale, il seguente:

configuration name: nome della configurazione corrente;

chip name: nome in codice dell'oggetto se esistono più oggetti in una classe; arbitrario se l'insieme contiene un unico dispositivo;

type: classe di appartenenza dell'oggetto;

identif: codice numerico che identifica lo specifico oggetto nell'ambito di una configurazione;

bus: identificatore di un oggetto bus con cui lo specifico oggetto può interagire in qualche modo;

address: indirizzi fisici compresi nello spazio indirizzi visto dallo specifico oggetto;

com1…com4: quattro campi che codificano particolari caratteristiche di uno specifico oggetto.

Si riportano di seguito le descrizioni degli oggetti attualmente utilizzabili nelle configurazioni di ASIM. Ciascuno di essi è descritto mediante una scheda sintetica di configurazione come oggetto, la descrizione funzionale e il modello di programmazione. Nel capitolo successivo sono presentati concreti esempi di configurazione di sistemi ASIM in cui sono impiegati i componenti qui descritti. Gli esempi sono corredati di programmi scritti nel linguaggio assembler del processore Motorola MC 68000 da utilizzare per il test delle configurazioni sviluppate.

4.1 Oggetti ASIM

4.1.1 Il dispositivo Processore

Un processore in ASIM è descritto mediante le seguenti informazioni codificate:

Name	Nome dell'oggetto processore (chiave) {M68000,...}
Identif	Intero che identifica univocamente l'oggetto CPU in una configurazione
Type	Identificatore dell'insieme di appartenenza dello specifico oggetto
Address1	Inizializzazione User Stack Pointer (USP)
Address2	Inizializzazione Supervisor Stack Pointer (SSP)
BUS	Identificatore del bus cui è connesso il processore
COM1	n.s.
COM2	n.s.
COM3	n.s.
COM4	n.s.

Tabella 1

Allo stato, l'unico processore supportato in Asim è il processore MC68000. L'ambiente di sviluppo è fornito di un assemblatore assoluto (asm.exe utilizzabile anche da una finestra MSDOS) che produce un file eseguibile (immagine di memoria) ed uno di listing utilizzato da ASIM durante la fase di debugging. Per quel che riguarda le caratteristiche del processore MC68000 e del suo linguaggio assembler si rimanda ai manuali Motorola riportati in bibliografia e al manuale assembler fornito con l'ambiente ASIMtool.

L'assemblatore, a partire da un file simbolico (*.a68) genera un assoluto (*.h68) in linguaggio macchina in formato S-File di Motorola e un file di listing (*.lis) utilizzato anche dal simulatore per effettuare il trace simbolico del programma durante il processo di simulazione.

Quando la finestra attiva è quella associata ad un MC68000, tra le varie voci di menù compare anche quella *Processore* grazie alla quale è possibile accedere a tutti i comandi precedentemente descritti.

Nella finestra del processore (mediante l'attivazione di *Mostra Registri*) è possibile visualizzare, oltre a tutti i registri interni, anche il numero di clock macchina consumati per eseguire tutte le istruzioni dall'inizio fino al punto corrente della simulazione (i valori dei cicli di clock impiegati per eseguire ciascun istruzione sono stati tratti dal manuale del processore. Sebbene questo numero non tenga conto di eventuali cicli di attesa per l'accesso in memoria, moltiplicandolo per il periodo di clock della macchina reale che si sta simulando, si ottiene una buona stima dei tempi di elaborazione "reali" di un dato programma).

Per inserire in una configurazione un MC68000 occorre selezionare dal menù *Configura* il comando *Aggiungi CPU* e specificare nella finestra di dialogo (Fig. 11), nel campo Name il valore "M68000"; nel campo "Address 1" l'indirizzo del puntatore allo stack utente; nel campo "Address 2" l'indirizzo del puntatore allo stack supervisore; nel campo in "BUS" l'identificatore del bus o memoria cui è connesso il processore.

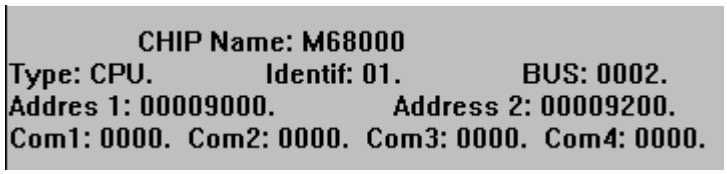


Fig. 11 Finestra di configurazione del processore M68000

I campi Com1..Com4 possono essere lasciati al valore zero non avendo per tale componente alcun significato.

Si osservi che il campo address2 contenete SSP (il puntatore allo stack supervisore) deve essere inizializzato con una locazione fisica di memoria RAM del processore; se così non fosse, in seguito ad un'operazione di accesso a tale indirizzo, verrebbe generato un "bus error" a cui il processore reagirebbe con il tentativo di salvataggio di alcuni registri sullo stack supervisore innescando, quindi, una condizione di stallo da cui non sarebbe possibile uscire se non con una operazione di "reset" del processore. Eventi anomali come "bus error", "istruzione illegale", "violazione di privilegio" vengono sempre segnalate.

Il parametro "BUS" fissa il collegamento del processore al bus di sistema mediante cui accedere agli altri componenti di una configurazione. Il parametro “Identif” è utilizzato dagli oggetti (device) della specifica configurazione per riferire il processore relativamente all’eventuale linea di interruzione ad esso connessa.

Un device può essere connesso ad un dispositivo in grado di gestire un segnale di interruzione, quale può essere un processore o un priority interrup controller (PIC). La connessione avviene su una linea di interruzione (ASIM ne mette gestisce al massimo quindici, da 1 a F esadecimale. Il 68000 ne gestisce solo sette (ad esso non possono essere connesse, pertanto, linee di interruzione oltre la settima). Le richieste di interruzione vengono servite in maniera dipendente dal dispositivo. Il processore MC68000, ad esempio, opera su 7 livelli prioritari e privilegia quelle di livello superiore al livello corrente indicato nel registro di stato.

Oltre alle linee di interruzione ASIM mette a disposizione il supporto per la gestione delle interruzioni vettorizzate così come implementate nel processore 68000. Il processore, all’atto dell’arrivo di una interruzione (Hw o Sw), avvia un ciclo di bus mediante il quale acquisisce, da parte della periferica interrompente, uno specifico numero di vettore, espresso su 8 bit (l’area ROM dei vettori occupa il primo k dello spazio indirizzi di memoria). Tale numero, moltiplicato per 4 mediante shift a sinistra di 2 posizioni (2^8<<4=1K), punta ad una locazione di ROM contenente il vettore associato alla Interrupt Service Routine (puntatore alla ISR) e a cui il processore salta dopo aver salvato il contesto globale dell’elaborazione (registro di stato e PC). Tutti i vettori sono di 32 bit (espressi su due word) e risiedono in area dati dello stato supervisore. Fa eccezione il primo, il vettore #0, di reset che è di 4 word, di cui 32 bit per inizializzare il PC e 32 il supervisor stack pointer, e risiede in area programmi. I 256 vettori di interruzione sono numerati da 0 a FF esadecimale. Per il 68000, i vettori delle interruzioni dal numero 12 al 23 e dal numero 23 al 48, sono riservati per future espansioni del processore, i vettori da 0 a 12, sono dedicati al processore, i 7 vettori 25-31, operano secondo il meccansimo degli pseudovettori (i vettori sono assegnati in modo fisso alle linee di interruzione), sono detti **autovettori**, hanno un livello prioritario da 1 a 7 e sono stati introdotti nel 68000 per supportare le periferiche operanti con la serie di processori a 8 bit 68xx. Essi non richiedendo l’esecuzione del ciclo di bus per leggere dalla periferica il numero di vettore, vanno inizializzati direttamente nella configurazione di ASIM in base al tipo di interruzione. Ciò deve essere fatto, ad esempio, per i device Terminal e PIA che non supportano vettori ma autovettori e non per il PIC che fornisce il vettore.

Per la gestione delle interruzioni ASIM offre la possibilità di associare, ad un singolo evento interrompente, un meccanismo semplice di gestione prioritaria che, nell’ambito di una stessa linea di interruzione, consente di selezionare, fra più interruzioni simultaneamente presenti, quella a maggiore priorità. La scelta prioritaria simula di fatto un meccanismo hardware con sedici livelli di priorità per linea (da $0 a $F, con 0 a maggior priorità). Si faccia attenzione a non confondere tale meccanismo con lo schema di gestione prioritaria del processore 68000 in quanto, quest’ultimo, come detto, implementa una gestione a 7 livelli (con il 7 livello a maggiore prioritarità) nell’ambito di 7 segnali di interruzione codificati mediante le tre linee IPL0-IPL3 a cui corrispondono altrettanti livelli pripritari. Il livello 0 è usato per indicare la maschera totale delle interruzioni, il 7 le NMI.

Per programmare un gestore (handler) di un device che ricorre alle interruzioni vettorizzate, vanno, per questo, specificati, nella pagina di configurazione, i tre parametri: linea, vettore e priorità. Un evento interruzione è, pertanto, caratterizzato in ASIM da un descrittore contenete tre campi {*linea d'interruzione* bit c0-c1, *priorità* bit c2-c3 e *vector number* bit c4-c7}. Se si usano gli autovettori, il campo vector number non va specificato in quanto associato al campo linea interruzione che può assumere i valori 1-7.

5 Il dispositivo Bus/Memoria

I campi relativi alla configurazione di un bus sono così codificati:

Name	Nome mem/bus (a solo scopo descrittivo);
Identif	Intero che identifica univocamente l'oggetto in una configurazione;
Type	Identificatore dell'insieme di appartenenza dello specifico oggetto;
Address1	Indirizzo base RAM;
Address2	Indirizzo base ROM;
BUS	Identificatore di bus esterno a cui rendere visibile le memorie di bus/mem;
COM1	Identificatore di bus esterno da cui importare la visibilità degli oggetti ad esso connessi;
COM2	dimensione (in esadecimale) della memoria RAM in blocchi da 4 Kbyte;
COM3	dimensione (in esadecimale) della memoria ROM in blocchi da 4 Kbyte;
COM4	n.s.

Tabella 2

La classe Bus/Memoria è costituita da un unico oggetto bus. In un sistema hw il bus consente di effettuare lo scambio di dati e controlli fra un mittente ed un destinatario (mediante opportuni cicli battuti da un master del bus) In ASIM il bus, oltre a simulare un dispositivo in grado di emulare il comportamento di un bus fisico e consentire lo scambio dati tra oggetti, consente di collegare, secondo differenti schemi, oggetti appartenenti a differenti sistemi all'interno della stessa configurazione. In particolare, il bus ASIM è anche "sostegno" per la memoria (ROM/RAM), che essendo intrinsecamente un oggetto passivo (non dotato di capacità di sviluppare da sola cicli di bus), viene aggregata a tale dispositivo per consentirne l'accesso. Un bus può anche essere configurato senza memoria. In tal caso esso viene utilizzato per interfacciare altri oggetti (bus, memorie appartenenti ad altri bus, processori, ecc.).

Il dispositivo Bus/Memoria può essere configurato, pertanto, nei seguenti modi:

- semplice bus da utilizzare per connettervi dispositivi di tipo qualsiasi;
- modulo di memoria RAM e/o ROM
- bus con memoria RAM e/o ROM
- bus con interfaccia verso un altro bus con o senza memoria RAM e/o ROM.

Il tipo di funzione assegnata dipende, ovviamente, dai valori assegnati ai parametri che compaiono nella finestra di dialogo "Aggiungi MMU/BUS". In particolare, il parametro Nome va posto a "MEMORIA"; l'identificatore va specificato secondo le regole discusse precedentemente ed il suo valore è quello che verrà utilizzato nel definire il "BUS" per ogni dispositivo che deve essere connesso a questo.

La memoria RAM o ROM configurabile ha parallelismo byte e dimensione in K espressa in esadecimale. In particolare, in Address1 (e Address2) va posto l'indirizzo base della memoria RAM (ROM) se presente mentre in COM2 (e COM2) la sua dimensione. Se la RAM (ROM) non è presente COM2 (COM3) va posto a zero (si faccia attenzione a non superare la dimensione di $8000 che corrisponde a 32 M e a che gli indirizzi di RAM e ROM non si sovrappongano). Il parametro "Com4" non svolge alcuna funzione e può essere lasciato al valore nullo.

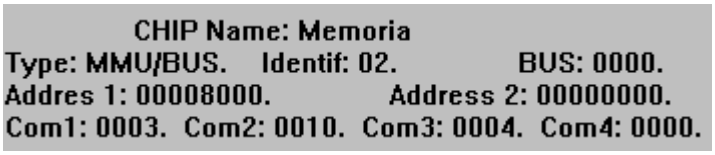


Fig. 12-Finestra di configurazione del BUS/Memoria

Il campo BUS può contenere l'identificativo di un bus esterno. Ciò serve, ad esempio per configurare un modulo di sola memoria (RAM e/o ROM) connessa al bus esterno il cui identificativo è posto nel campo BUS.

Ad esempio in fig.13 è riportata una configurazione in cui un dispositivo memoria privo di bus è reso visibile, come spazio di indirizzamento, al bus con id=1 che interconnette una CPU, una RAM ed una ROM.

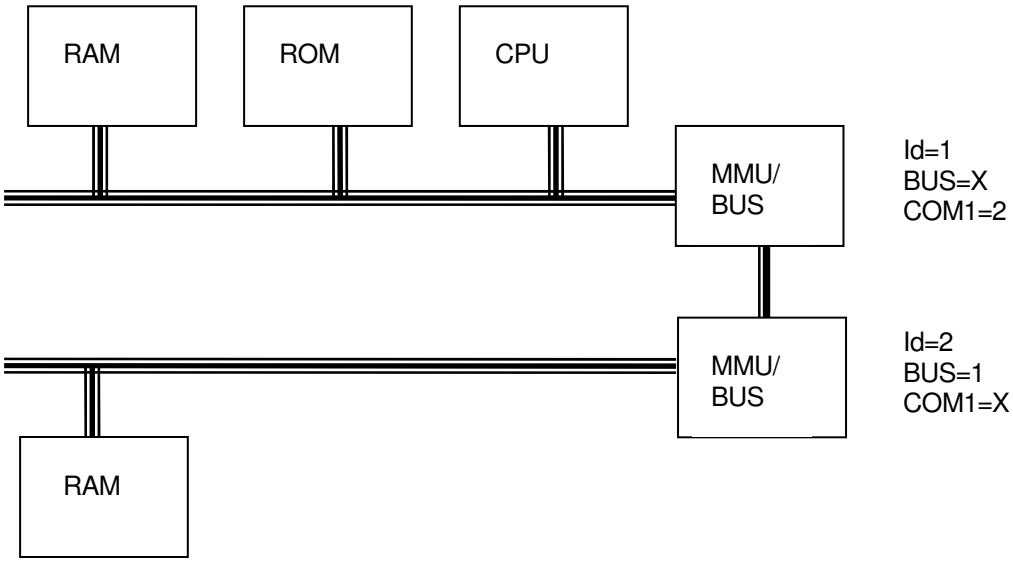


Fig. 13-Esempio di visibilità del dispositivo MMU/BUS

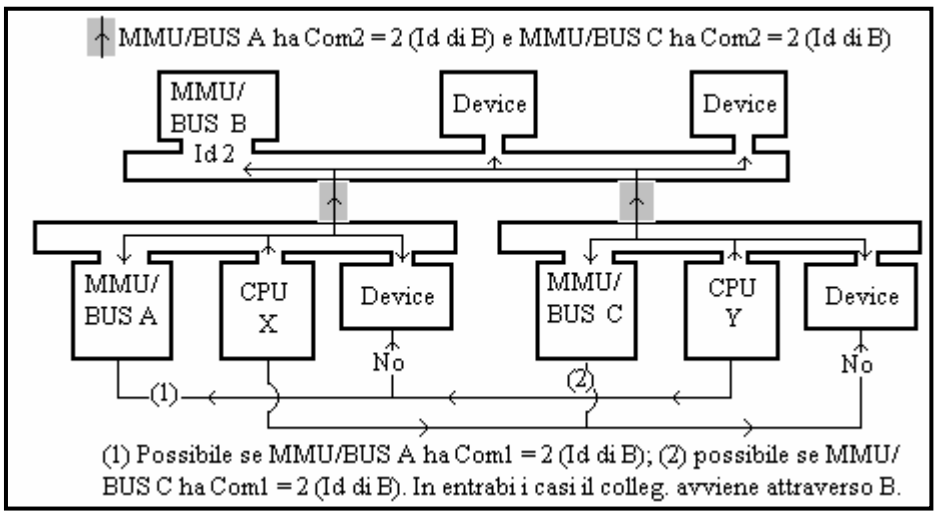


Fig. 14-Esempio di visibilità del dispositivo MMU/BUS

Se quello che si è definito è un bus A (con o senza memorie), assegnando a "BUS" il valore di un Identificatore di un altro bus B, si realizza una connessione tra i due in modo tale che un eventuale processore connesso al bus B "vede" anche (se ci sono) le memorie RAM e ROM di A (ma non eventali dispositivi connessi al bus A); ciò ovviamente nell'ipotesi che gli insiemi di indirizzi di queste memorie siano disgiunti da quelli di memorie e dispositivi connessi al bus B (Fig. 13 e Fig. 14).

Il dispositivo 1TO4BusInterface

I campi relativi alla configurazione di un bus sono così codificati:

Name	Nome del bus a cui l'oggetto è connesso (opzionale);
Type	Identificatore assoluto nella configurazione;
Address1	Valore iniziale dello spazio indirizzi gestito dall'oggetto;
Address2	Valore finale dello spazio indirizzi gestito;
BUS	Identificatore di bus esterno connesso all'ingresso;
COM1	Identificatore di bus esterno connesso all'uscita 1, se 0 la linea è n.c.;
COM2	Identificatore di bus esterno connesso all'uscita 2, se 0 la linea è n.c.;
COM3	Identificatore di bus esterno connesso all'uscita 3, se 0 la linea è n.c.;
COM4	Identificatore di bus esterno connesso all'uscita 4, se 0 la linea è n.c.;

Tabella 3

L'oggetto 1TO4BusInterface è un multiplexer di bus bidirezionale (1 ingresso-4 uscite) in grado di espandere le possibilità di interconnessione dell'oggetto bus/mem.

La selezione della via su cui instradare la connessione è fatta in base all'appartenenza dell'indirizzo presente sul bus in ingresso ad un intervallo di indirizzi fisici definiti mediante i parametri posti nei campi COMi. Esso, pertanto, è in grado di instradare richieste di accesso, poste in ingresso, verso bus connessi in uscita, selezionando la destinazione in base all'indirizzo.

Con 1TO4BusInterface possono realizzarsi ad esempio i seguenti schemi:

a. da un processore verso alcuni bus (da 1 a 4 bus);

b. da più processori verso alcuni bus (da 1 a 4 bus);

c. da un bus verso alcuni bus (da 1 a 4 bus);

d. da uno o più processori verso alcuni 1to4BUSINT (da 1 a 4).

Il tipo di collegamento d) indica la possibilità di costruire cascate di questi dispositivi, rendendo praticamente illimitato il numero di bus cui può accedere un processore.

Come ogni dispositivo 1to4BUSINT può essere connesso ad un bus, realizzando così collegamenti di tipo c); questo tipo di collegamento è simile a quello descritto nel paragrafo precedente, ma differisce da questo perché la connessione può essere verso più di un bus.

Uno o più processori possono essere connessi a più bus anche connettendoli ad un bus cui sono connessi uno o più dispositivi del tipo in esame oppure connettendoli direttamente ad un 1to4BUSINT.

Poiché tutti i tipi di connessioni viste sono combinabili tra loro, è possibile realizzare un insieme molto ampio di configurazioni ed, in particolare, strutture di "crossbar".

Il dispositivo non ha registri interni. Il menù *Device* associato a questo dispositivo è quindi diverso da quello presentato; l'unico comando disponibile è *Trace Accessi* che consente di visualizzare le richieste di accesso: richieste di lettura , scrittura e modifica (ciclo di lettura+scrittura utilizzato tipicamente dai sistemi operativi per

implementare semafori). Quando si seleziona il comando, le richieste iniziano ad essere registrate e visualizzate fin quando non si invoca nuovamente *Trace Accessi*; in ogni momento sono memorizzate al più le ultime trenta richieste e, per visualizzarle, basta far scorrere verticalmente la finestra con l'apposita barra di scorrimento.

La definizione dei parametri avviene attraverso la finestra di dialogo cui si accede dal menù *Configura* con il comando *Aggiungi Device*. Il "Nome" da indicare; per il campo Identificatore è 1TO4BUSINT.

"Indirizzo1" e Indirizzo 2" delimitano l'intervallo di indirizzi ammessi in ingresso: quando arriva una richiesta con indirizzo compreso in detto intervallo, il dispositivo cerca di instradarla verso l'uscita appropriata; se il tentativo fallisce, restituisce al richiedente un "Bus error".

Gli identificatori dei bus o altri 1to4BUSINT connessi a valle vanno indicati come segue:

- Identificatore del bus da connettere alla linea di uscita 1 in "Com1";
- Identificatore del bus da connettere alla linea di uscita 2 in "Com2";
- Identificatore del bus da connettere alla linea di uscita 3 in "Com3";
- Identificatore del bus da connettere alla linea di uscita 4 in "Com4";

Se non si vuole collegare una linea di uscita ad alcun bus si deve lasciare il parametro corrispondente a zero.

La scelta dell'insieme di indirizzi da far corrispondere ad ogni elemento viene effettuata da parte del programma come mostrato in Fig. 15:

Com1	Com2	Com 3	Com4	BUS int 1	BUS int 2	BUS int 3	BUS int 4	N
0	0	0	0	nc	nc	nc	nc	-
p	0	0	0	c	nc	nc	nc	1
x	p	0	0	c/nc	c	nc	nc	2
x	x	p	0	c/nc	c/nc	c	nc	3
x	x	x	p	c/nc	c/nc	c/nc	c	4

N numero di parti in cui è diviso l' intervallo di indirizzi; p positivo; x indifferente (0 o p); c connesso; nc non connesso; c/nc connesso se il corrispondente Com è positivo, altrimenti non connesso.

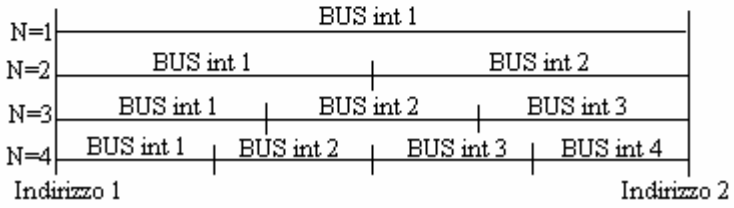


Fig. 15-Tabella scelta intervallo indirizzi

Se uno degli identificatori, precedenti l'ultimo non nullo, non è stato definito, lasciandolo a 00, una richiesta con indirizzo nell'intervallo ad esso corrispondente genererà un "Bus error".

Riepilogando l'intervallo di indirizzi I=(address2-address1) viene diviso in un numero n di parti uguali al numero di connessioni in uscita attive. Esso deve essere, pertanto, definito in modo da essere un multiplo di n.

Non è possibile definire intervalli variabili all'interno di I da assegnare alle singole connessioni in uscita. Conviene assegnare le connessioni in uscita, se presenti, a partire dalla 1 e via via fino alla 4. Assegnata una connessione i delle n previste, resta fissato, comunque, il sotto intervallo di indirizzi a questa associato (le connessioni ad essa inferiori possono anche non essere assegnate). I valori di detti intervalli sono esprimibili mediante la formula:

```
I=address1-adress2
Indirizzo di inizio=(address1 + (i-1)*I/n)
Indirizzo di fine= (address1 + i*I/n) -1
```

Il metodo di scelta descritto consente di combinare in modo molto vario gli insiemi di indirizzi corrispondenti a ciascuna linea di uscita. Si osservi che ogni insieme di indirizzi deve essere tale da includere tutti quelli delle locazioni (di memoria o di device) effettivamente presenti sul corrispondente bus a valle; se così non fosse non ci sarebbe modo di accedere a quelle locazioni che, pur presenti sul bus, hanno un indirizzo fuori dell'intervallo. Nessun problema, invece, per indirizzi compresi nell'intervallo, ma senza corrispondente locazione sul bus; in tal caso è il bus che si preoccupa di far restituire al richiedente un "Bus error".

Se si vuol connettere il 1to4BUSINT ad un bus, il valore dell'Identificatore di tale bus va in "BUS".

Il dispositivo è dotato della capacità di monitorare l'attività di accesso su uno dei bus (sia in ingresso che in uscita). Ad esempio 1to4BUSINT può essere utilizzato con una sola connessione in uscita al fine di monitorare gli accessi su un bus da parte di un processore. E' anche possibile assegnare più linee di uscita allo stesso bus (specificando più volte lo stesso identificatore). Ciò è utile quando ad un bus sono connessi una memoria e dei dispositivi con indirizzi separati e gli indirizzi intermedi sono invece assegnati ai dispositivi di un altro bus.

6 Il dispositivo generatore di interruzioni 1to4INTGEN.

I campi relativi alla configurazione di un bus sono così codificati:

Name	Nome del bus a cui l'oggetto è connesso (opzionale);
Type	Identificatore assoluto nella configurazione;
Address1	Valore iniziale dello spazio indirizzi gestito dall'oggetto;
Address2	Valore finale dello spazio indirizzi gestito Indirizzo 1" + 13;
BUS	Identificatore di bus esterno connesso all'ingresso;
COM1	Identificatore dei dispositivi in grado di gestire le interruzioni, se 0 la linea è n.c.;
COM2	Identificatore dei dispositivi in grado di gestire le interruzioni, se 0 la linea è n.c.;
COM3	Identificatore dei dispositivi in grado di gestire le interruzioni, se 0 la linea è n.c.;
COM4	Identificatore dei dispositivi in grado di gestire le interruzioni, se 0 la linea è n.c.;

Tabella 4

Il dispositivo include un generatore di interruzioni programmabile e quattro timer. Esso può essere, ad esempio, impiegato per generare interruzioni ad intervalli di tempo prefissati (mediante i timer) e per definire meccanismi per inviare, a programma, segnali di interruzione ad un gestore di interruzioni.

Per connettere un 1to4INTGEN ad una macchina da simulare, occorre specificare i parametri nella finestra di dialogo, cui si accede dal menù *Configura* con il comando *Aggiungi Device*. "Nome elemento" è 1TO4INTGEN, "Indirizzo 1" deve essere pari e rappresenta l'indirizzo del registro I (indirizzo relativo 00); tutti gli altri registri hanno indirizzo dato da "Indirizzo 1" + indirizzo relativo (i valori dell'indirizzo relativo sono quelli di figura 10.b). "Indirizzo 2" deve essere uguale ad "Indirizzo 1" + 13 (numero esadecimale). In "BUS" va l'Identificatore del bus cui il dispositivo è connesso. I parametri "Com1", "Com2", "Com3" e "Com4" (corrispondenti, rispettivamente, alle linee 1, 2, 3 e 4) consentono di specificare le connessioni verso i dispositivi gestori delle interruzioni. Ad esempio "Com2=3" assegna la linea di uscita 2 al dispositivo gestore di interruzioni di idendificatore 3. Se una linea non è connesso ad alcun dispositivo il valore di com va posto a zero. Più linee possono essere connesse allo stesso dispositivo.

Il dispositivo 1to4INTGEN ha dieci registri r0-r9 a sedici bit di cui i primi due dedicati al generatore di interruzioni, i rimanenti 2x4 ai 4 timer. La Fig.16a mostra l'insieme dei registri e la loro funzione, la 16-b la finestra di stato-controllo associata al dispositivo e gli indirizzi corrispondenti a ciascun registro (gli indirizzi sono relativi all'indirizzo base).

L'accesso ai vari campi di controllo per la modifica è fatta accedendo ai vari campi dal menù di modifica dopo aver selezionato la finestra con la configurazione.

6.1.1 Il generatore di interruzioni

Il 1to4INTGEN può essere connesso con al massimo quattro dispositivi in grado di gestire interruzioni (processori o priority interrupt controller). Il registro I posto all'indirizzo relativo $0 del dispositivo contiene la struttura dell'evento interruzione che si vuole inviare. Un processore che vuole inviare un'interruzione ad un altro (o anche a se stesso) deve specificare il tipo di interruzione assegnando al registro I (indirizzo relativo 00) i valori della linea, priorità ed del vector number nell'ordine riportato in fig.16a (le due cifre esadecimali del byte meno significativo specificano linea e priorità; le due cifre esadecimali del byte più significativo specificano il vector number. Una volta definita l'interruzione, per inviarla, occorre scrivere un opportuno valore nel registro C (indirizzo relativo 02); di questo registro, rappresentato in fig.16a, sono utilizzati solo i quattro bit meno

significativi. Per inviare un'interruzione ad uno dei 4 dispositivi gestori di interruzione connettibili, deve essere settato ad 1 il bit i-esimo di C della linea interessata.

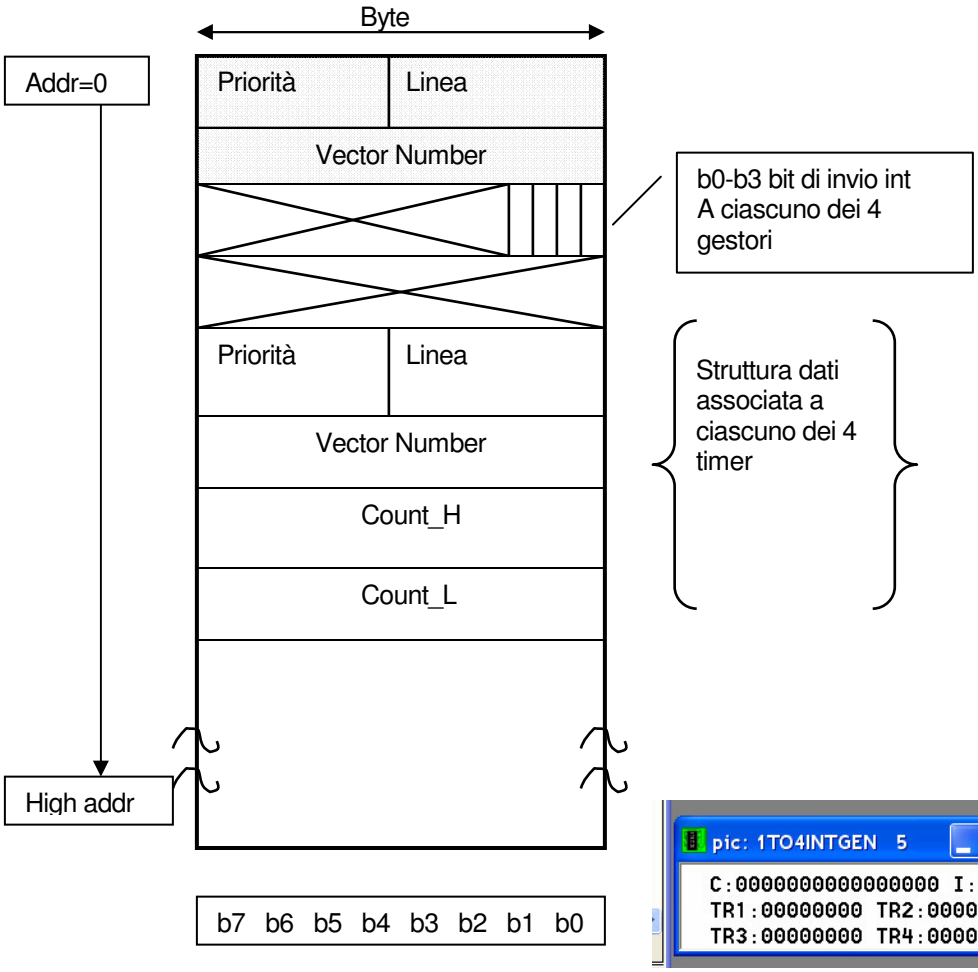


Fig. 16a – Modello 1to4INTGEN

Fig. 16b – Finestra di Stato

E' possibile inviare, contemporaneamente, la stessa interruzione anche a tutti e quattro i dispositivi connessi, ponendo ad 1 tutti e quattro i bit meno significativi di C. Se ad una delle linee di uscita non è connesso alcun gestore delle interruzioni, l'interruzione non viene inviata; se più di una linea è connessa allo stesso dispositivo e vengono posti ad 1 i corrispondenti bit di C, vengono inviate tante interruzioni dello stesso tipo al gestore, quante sono le linee ad esso connesse (attivate con bit 1 in C).

6.1.2 Il timer

I quattro timer TR1-TR4 sono identici e sono collegati rispettivamente a linee di uscita 1-4. A ciascuno di essi sono associate due word di cui la meno significativa per specificare l'interruzione (vale ancora quanto detto a proposito del registro I) e la più significativa per definisce il valore iniziale del contatore. Se V è la velocità del clock associato al dispositivo (di default V = 1, ma può essere modificata come detto in 3.2.4), ogni V colpi di clock il valore del contatore si decrementa di 1. Quando questi arriva a zero, viene inviata l'interruzione specificata al gestore di interruzioni connesso ed il contatore cessa di decrementarsi.

Poiché i dispositivi connessi alle linee di uscita del 1to4INTGEN non devono necessariamente essere diversi, è possibile associare allo stesso dispositivo anche tutti e quattro i timer.

7 IL Priority Interrupt Controller

I campi relativi alla configurazione del dispositivo sono così codificati:

Name	I8259PIC
Type	Identificatore assoluto nella configurazione;
Address1	Indirizzo base pari per il dispositivo
Address2	Indirizzo base+$13
BUS	Ident. del bus a cui il dispositivo è connesso
COM1	Ident. del dispositivo PIC o CPU gestore delle interruzioni
COM2	|vettore (VVVVxxxx)|priorità (xxxxPPxx)|liv. Int (xxxxxxLL)|
COM3	N.U.
COM4	N.U

Tabella 5

7.1.1 Configurazione ASIM del PIC

Per inserire questo componente in una configurazione di sistema, bisogna attenersi alla stessa procedura seguita per gli altri componenti di tipo **Device**.

I parametri presenti nella finestra **Aggiungi Device** permettono di gestire la connessione del gestore programmabile delle interruzioni con gli altri componenti della configurazione.

Nome Elemento è utilizzato per specificare il tipo di componente da inserire, nel caso in esame deve essere *"I8259PIC".*

Identificatore deve essere un numero *compreso tra 01 ed FF* e viene utilizzato dal programma per riferirsi a questo dispositivo.

Indirizzo 1 rappresenta l'indirizzo più basso per accedere al componente; in questo caso deve essere un *numero pari.*

Indirizzo 2 definisce l'indirizzo più alto per indirizzare il componente, nel caso in esame deve essere *uguale ad Indirizzo1 + 1.*

Questi ultimi due parametri permettono di definire per quali indirizzi il decodificatore di indirizzi attiva la linea **CS.**

BUS determina l'*Identificatore del bus* a cui è connesso il dispositivo. La connessione delle linee **bus dati**, **A0**, **RD** e **WR** del controllore delle interruzioni al processore è realizzata nel simulatore, scegliendo per **BUS** l'identificatore di un componente MMU/BUS, a cui è stato collegato il dispositivo di tipo CPU.

COM1 definisce l'*Identificatore del gestore delle interruzioni*, cioè il componente (di tipo processore o PIC) al quale trasmettere le interruzioni non mascherate.

COM2 i 2 byte di tale campo (vvpl) ospitano un descrittore di interruzione (vv=vector number, p=priorità e l=linea d'interruzione,). Poiché il vector number è generato dal PIC a partire dal valore base del vettore memorizzato in un suo registro e secondo uno spiazzamento dipendente dalla specifica interruzioni pervenute in una delle sue 8 linee di interruzione, il campo vector number (ultime due cifre esadecimali), in tal caso, non deve essere specificato.

COM3 e COM4 non utilizzati.

La finestra associata in ASIM al PIC è presentata in Fig.20.

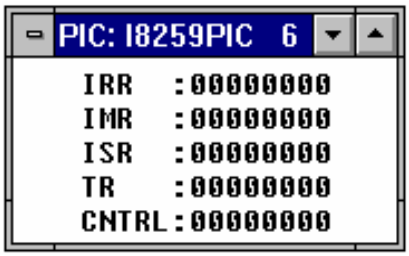


Fig. 17 – Finestra ASIM con i registri del PIC

Il PIC implementato è derivato, nella logica, dal componente industriale Intel 8259, ma presenta un numero di registri e di modi di funzionamento limitato rispetto ad esso. In Fig.18 è mostrato il modello di programmazione e le linee fisiche simulate. Tale dispositivo è stato progettato dall'Intel per la famiglia di processori 80x86 al fine di estendere le capacità di gestione delle interruzioni del processore. L'8259 è dotato di una sola linea di interruzione INTR, una linea di acknowledge INTA e gestisce fino a 8 linee di interruzione differenti, con livelli di priorità multipli (fino ad 8 livelli di priorità). Più dispositivi 8259A possono essere connessi in cascata (fino a 8 per la gestione di max 64 linee di interruzione). Accetta richieste di interruzioni dai dispositivi di I/O connessi alle sue linee, determina, a seconda dell'algoritmo di gestione prioritaria selezionato, quale delle interruzioni simultaneamente attive, ha la priorità più alta per il servizio, trasmette l'interruzione al processore assertando la linea INTR, attende che il processore risponda con l'acknowledge INTA, trasmette al processore il vettore dell'interruzione associato alla linea selezionata. A questo punto Il processore avvia il servizio dell'interruzione.

La gestione prioritaria si basa sui seguenti tre schemi programmabili per tramite di un'opportuna codifica del registro di controllo CNTRL:

- Fully nested: le richieste di interruzione sono ordinate secondo uno schema a priorità fissa che va da IR0 a IR9, con IR0 la linea più prioritaria;
- Round Robin: è uno schema prioritario a rotazione. La linea di interruzione più prioritaria servita diventa la meno prioritaria dopo il servizio;
- Maschera delle interruzioni: consente l'inibizione o l'abilitazione delle linee di interruzione a programma.

Il modo secondo cui può operare il PIC deve essere configurabile in fase di inizializzazione del dispositivo (dopo il suo reset) ma può essere cambiato anche dai programmi di gestione.

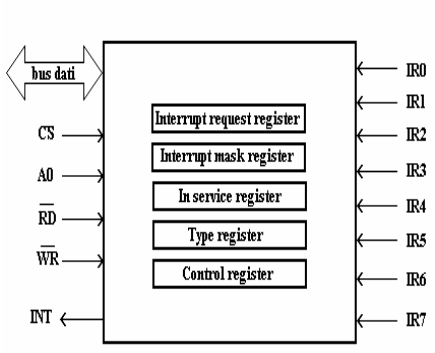


Fig. 18 - modello di programmazione del PIC

7.1.2 Modello di programmazione

Il dispositivo PIC simulato occupa due locazioni di memoria da cui indirizzare i 5 registri a 8 bit CNTRL, IRR, ISR, TR, IMR a disposizione del programmatore ed il cui significato è il seguente:

- Il registro CNTRL (Control Register) permettere di controllare il comportamento del dispositivo;
- Il registro IRR (Interrupt Request Register) memorizza i segnali di interruzione;
- Il registro ISR (In Service Register) presenta al gestore delle interruzioni i segnali interrompenti in modo ordinato rispetto alla maschera e alla priorità corrente;
- Il registro TR (Type Register) memorizza la parte base dell'indirizzo del vettore
- Il registro IMR (Interrupt Mask Register) permette di mascherare singolarmente le interruzioni.

Il banco di 8 registri da 1 bit, detti Interrupt Request Reg (**IRR**), riceve in ingresso le 8 linee di interruzione (IR0-IR7) provenienti dalle periferiche collegate memorizzandone il valore. IRR è posto in cascata con un circuito per la gestione prioritaria delle interruzioni e con un altro banco di 8 registri da 1 bit, detto In Service Reg (**ISR**) in cui sono memorizzati solo i segnali di interruzione da servire. Fra questi è selezionato quello più prioritario. Il blocco addetto alla gestione prioritaria determina le priorità dei bit settati in IRR. Il bit più prioritario (la linea a priorità maggiore è la linea 0; la priorità decresce fino alla linea 7) viene selezionato e scritto nel registro ISR. Non possono essere attivate interruzioni in arrivo su linee meno prioritarie di quelle in cui è settato uno dei bit di ISR. Qualora sia abilitata l'apposito flag nel registro CNTRL, i bit di ISR sono cancellati automaticamente all'atto del servizio; in caso contrario sarà compito della routine che serve l'interruzione cancellare lo specifico bit mediante il registro CNTRL.

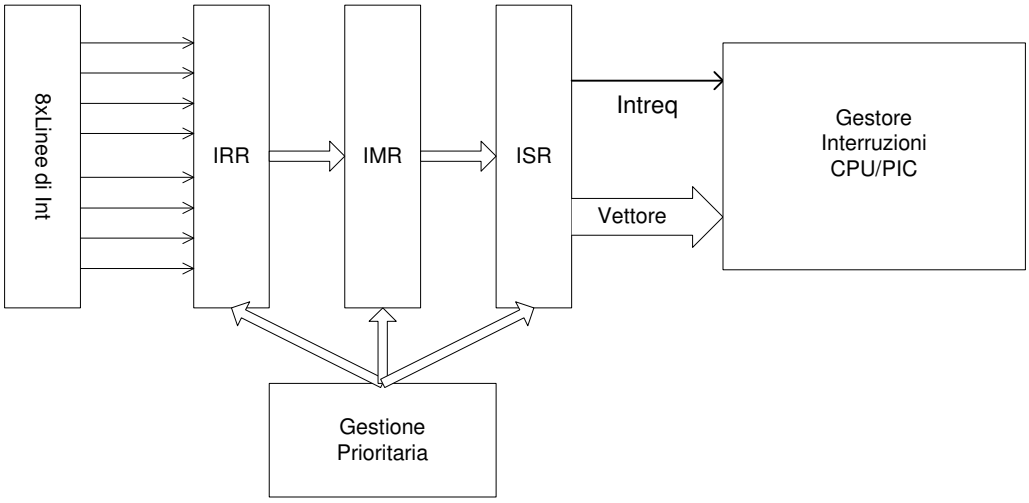


Fig.20 – Modello funzionale e di programmazione del PIC

Il registro Interrupt Mask (**IMR**) di 8 bit, serve da maschera delle interruzioni. I bit di mascheramento corrispondono in posizione a quelli del registro IRR; se il bit è settato ad 1 la corrispettiva linea di interruzione è mascherata. Il registro Type Register (**TR**) di 8 bit memorizza (nel formato aaaaannn) nei 5 bit più significativi (in posizione aaaaa), il valore base del vector number (compreso tra 0 e 2**5=32); nei successivi 3 bit meno significativi (nnn) la specifica linea interrompente settata automaticamente in base alla linea interrompente (ad

esempio se è interessata la linea 5 i 3 bit saranno automaticamente settati ad 101). Pertanto, tutte le interruzioni associate ad uno specifico PIC presentano vettori consecutivi composti da un unico indirizzo base di 5 bit ed uno spiazzamento di 3.

Una richiesta d'interruzione che arriva sulla linea i-esima, pone ad 1 il relativo bit del registro **IRR** e, se non mascherata e selezionata, è spedita al gestore delle interruzioni collegato al PIC (una CPU o un altro PIC). Il PIC trasmette, quindi, il relativo vettore (nel formato aaaaannn). Il bit i-esimo di IRR è automaticamente cancellato per rimuovere la causa d'interruzione.

Tra tutte le interruzioni che arrivano da altri device sulle linee di richiesta IR0-IR7 del componente, solo quella con priorità maggiore e non mascherata sarà da esso scelta. Se su una linea sono collegati più device che generano interruzioni contemporaneamente, viene scelta quello a priorità maggiore. ASIM opera secondo un meccanismo che simula un circuito Daisy Chain con 16 livelli di priorità per linea; il valore del livello prioritario viene spedito al PIC insieme al numero della linea.

Selezionata l'interruzione con la più alta priorità non mascherata, il dispositivo modifica i tre bit meno significativi del registro **TR** per generare lo specifico vettore relativo alla linea di interruzione interessata da spedire al processore.

Dopo aver trasmesso al processore l'interruzione sulla linea i-esima, il bit i-esimo di **ISR** viene posto ad 1, il corrispondente di **IRR** viene posto a 0. Se il bit **AEOI** del registro **CNTRL** (bit 4), è pari ad 1, allora anche il bit i di **ISR** viene posto a 0 altrimenti, sarà compito della routine che serve l'interruzione annullarlo, per consentire il futuro servizio di interruzioni di livello prioritario pari o inferiore. Tale operazione va fatta settando opportunamente il valore di i nei tre bit meno significativi di **CNTRL** e ponendo il bit 3 **EOI** ad 1.

bit	significato
0,1,2	determinano il bit n da cancellare in **ISR**
3	Bit **EOI** (End Of Interrupt), se posto ad 1 fa cancellare il bit n-esimo del registro **ISR** puntato dai bit b0-b2
4	Bit **AEOI** (Automatic End Of Interrupt), se posto ad 1 fa cancellare automaticamente il bit in **ISR** dopo la trasmissione dell'interruzione
5	non utilizzato
6	Bit **RIS** (Register Interrupt Selector) seleziona negli accessi in lettura e se **RR**=1, il registro **ISR** se è posto a 1 il registro **IRR** se a 0
7	Bit **RR**, permette se posto ad 1 la lettura dei registri **ISR** o **IRR**

Tabella 6 - Significato dei bit di CNTRL

7.1.2.1 Modalità di selezione dei registri

I registri TR ed IMR condividono lo stesso indirizzo dispari (bit0=1). Al reset è preselezionato il registro TR che può essere acceduto in sola scrittura, i successivi accessi a tale indirizzo sia in lettura che scrittura selezioneranno il registro IMR. L'accesso all'indirizzo pari del dispositivo (bit0=0) riferisce in scrittura il registro di controllo CNTRL, in lettura, il registro di controllo CNTRL se il bit7 è posto a 0, i due registri IRR e ISR se posto ad 1. La scelta fra tali ultimi due registri è fatta in base al valore del bit 6 che se posto a 1 seleziona il registro ISR se a 0 il registro IRR. In tab.6 è riportata la struttura del registro di CNTRL, in tab.7 il riepilogo della selezione dei registri sopra descritta.

Indirizzo	Tipo di accesso	RR	RS	Registro
Pari	W	0/1	0/1	**CNTRL**
	R	0	0/1	
Pari	R	1	0	**IRR**
Pari	R	1	1	**ISR**
Dispari	W	0/1	0/1	**TR***
Dispari	W/R	0/1	0/1	**IMR**

** accessibile solo quando il dispositivo viene resettato*

Tabella 7-indirizzamento dei registri del gestore programmabile delle interruzioni

7.1.3 Collegamento del dispositivo con altri componenti di ASIM

I dispositivi a cui può collegarsi un PIC simulato in una configurazione sono:

- un processore, che avendo accesso ai registri del dispositivo, attraverso il sistema bus, permette la programmazione del componente;
- dei device di cui si vogliono gestire le interruzioni inviate;
- un eventuale PIC(o lo stesso processore) per gestire l'interruzione inviata dal dispositivo.

Le linee che collegano il gestore delle interruzioni al processore sono mostrate sulla sinistra dello schema in fig18.

. Il **bus dati** è utilizzato dal processore per il trasferimento dei dati da e verso il componente. La linea **CS** è utilizzata per la selezione del dispositivo; i registri interni sono selezionati dal bit meno significativo del bus indirizzo: **A0**, dai segnali di lettura-scrittura: **RD** e **WR**, oltre che dallo stato di taluni bit del registro di controllo. La linea d'interruzione **INT** trasmette al processore, o all'eventuale PIC, la richiesta d'interruzione selezionata dal dispositivo, tra quelle presenti in ingresso. Il processore reale impiega un segnale di **intack** per avviare un ciclo di bus in cui la periferica fornisce il proprio vector number. Tale segnale non è simulato in ASIM in quanto non viene gestito un simile ciclo di bus. ASIM sopperisce a ciò con un meccanismo proprio che consente ad un oggetto di inviare un messaggio su 16 bit con il valore della tripla vettore-priorità-linea precedentemente discussa.

Un esempio di collegamento del controllore programmabile delle interruzioni al processore è mostrato in Fig. 19 dove sono presentate solo le linee del processore Motorola 68000 coinvolte nel collegamento.

Sulla destra dello schema sono mostrate le linee che collegano il PIC ad un device di I/O, 8 linee delle interruzioni a priorità decrescente da **IR0** a**IR7**.

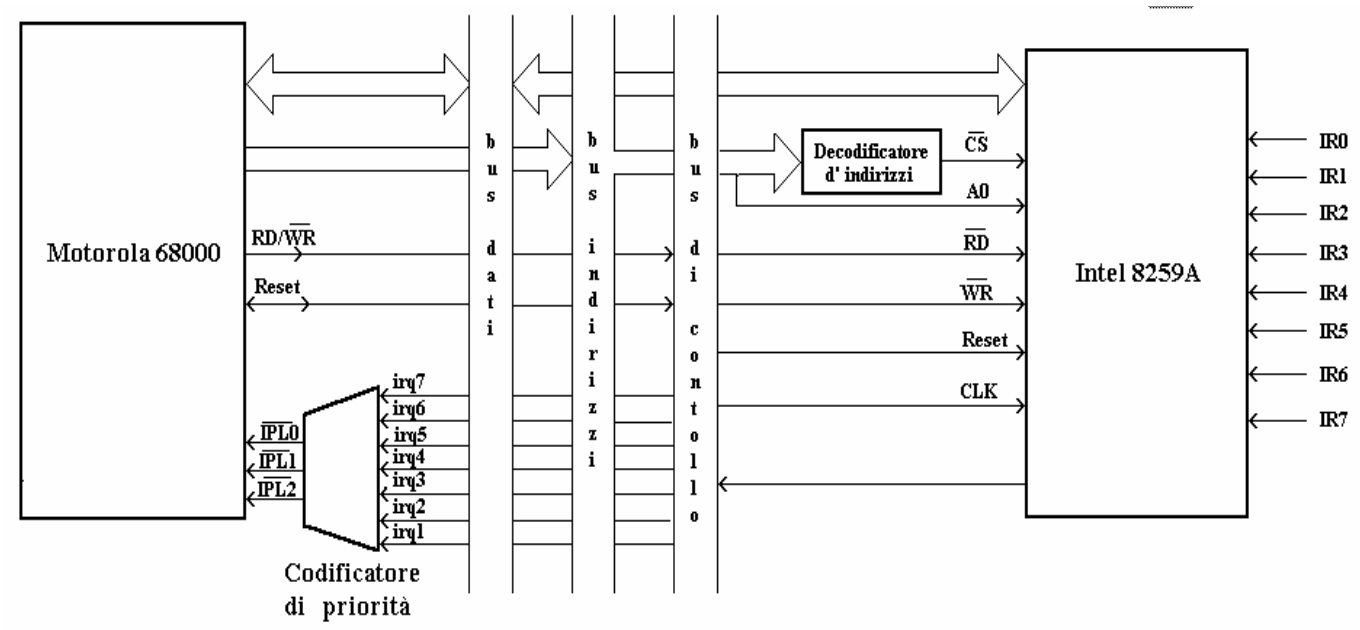


Fig. 19-Collegamento del PIC al processore MC68000

7.1.4 Programmazione del PIC

L'uso del PIC è riportato nell'esempio PIC fornito con ASIM. Esso si basa sulla configurazione, di seguito riportata, che simula un sistema composto da un processore 68000, una RAM una ROM con inizializzati 8 vettori di interruzione, un PIC e un 1to4intgen per la generazione delle interruzioni a programma, impiegando la sezione apposita e mediante uno dei 4 timer.

```
Configuration name: pic.cfg
CHIP Name: MEMORY
Type: MMU/BUS. Identif: 01. BUS: 0000.
Addres 1: 00008000. Address 2: 00000000.
Com1: 0000. Com2: 0010. Com3: 0008. Com4: 0000.
CHIP Name: M68000
Type: CPU. Identif: 02. BUS: 0001.
Addres 1: 00009000. Address 2: 00009200.
Com1: 0000. Com2: 0000. Com3: 0000. Com4: 0000.
CHIP Name: TERMINAL
Type: Device. Identif: 03. BUS: 0001.
Addres 1: 00002000. Address 2: 00002001.
Com1: 0006. Com2: 0006. Com3: 0007. Com4: 0000.
CHIP Name: 1TO4INTGEN
Type: Device. Identif: 04. BUS: 0001.
Addres 1: 00002004. Address 2: 00002017.
Com1: 0006. Com2: 0006. Com3: 0006. Com4: 0006.
CHIP Name: 1TO4INTGEN
Type: Device. Identif: 05. BUS: 0001.
Addres 1: 00002018. Address 2: 0000202B.
Com1: 0006. Com2: 0006. Com3: 0006. Com4: 0006.
CHIP Name: I8259PIC
Type: Device. Identif: 06. BUS: 0001.
Addres 1: 0000202C. Address 2: 0000202D.
Com1: 0002. Com2: 0001. Com3: 0000. Com4: 0000.
```

L'esempio riportato in fig.20 mostra una configurazione composta da un processore 68000, un modulo di memoria RAM di 16 Kbyte e ROM di 1Kbyte inizializzata con vettori di interruzioni, un dispositivo Terminal, due generatori di interruzioni e un PIC.

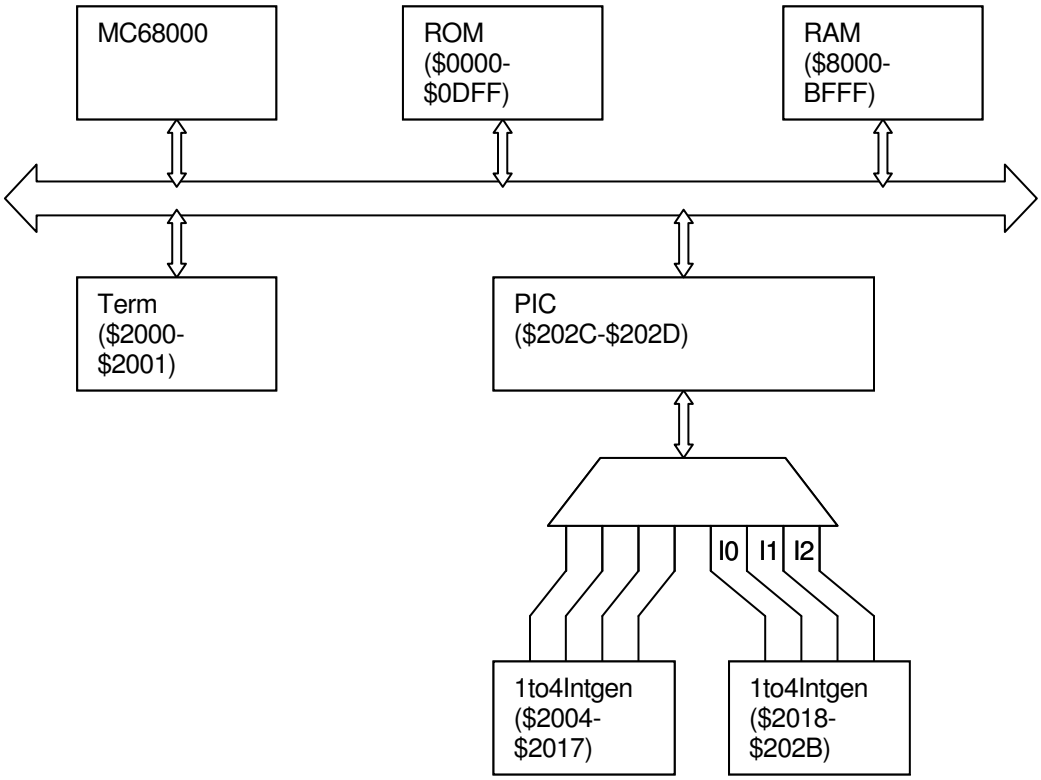


Fig.20 – Architettura utilizzata per l'esempio PIC

L'inizializzazione del PIC (a segutio di un Reset) viene effettuata scrivendo

- nel registro **TR** il valore base del vector number da trasmettere al processore mediante un accesso in scrittura all'indirizzo dispari; dopo tale scrittura tutti gli accessi in scrittura all'indirizzo dispari selezioneranno sempre il registro **IMR**.

- all'indirizzo pari nel registro **CNTRL** un valore tale da definire il modo in cui viene cancellato il bit in **ISR** (cioè in modo automatico o dalla routine che serve l'interruzione). Come detto, se la cancellazione del bit in **ISR** non è automatica, è compito della ISR cancellarlo scrivendo un byte in CNTRL che contenga nei tre bit meno significativi il numero del bit da cancellare ed il valore 1 nel bit 3.

- il registro **IMR** all'indirizzo dispari per, eventualmente, mascherare le singole linee di richieste di interruzione.

Si riporta di seguito un esempio di una sequenza di inizializzazione del PIC in cui in A0 è posto l'indirizzo base del device:

```
move.b      #$40,1(A0)
move.b      #$10,(A0)
move.b      #$F0,1(A0)
```

La prima istruzione scrive il vector number $40 in **TR**; la seconda seleziona in CNTRL la cancellazione automatica del bit in ISR; l'ultima setta la maschera delle interruzioni per le linee da 4 a 7.

Il dispositivo 1to4intgen presenta, come detto, al suo interno due componenti “generatori di interruzioni” programmabili. Ciascuno di essi permette di gestire 4 interruzioni verso un processore o PIC cui è collegato. Ciascuna linea di interruzione è caratterizzata dai tre parametri {vettore int (0-255), liv. prioritario (Irq0-Irq7), linea int (Ir0-Ir7)} e da un flag che se ad 1 attiva l’interruzione.

Il dispositivo contiene anche 4 timer indipendenti in grado di generare eventi interruzioni. Ciascun timer è programmabile mediante una coppia di registri di cui quello ad indirizzo più basso serve a specificare la linea di interruzione, così come descritto sopra, quello ad indirizzo più alto il valore di un contatore a decrescere che quando raggiunge lo zero genera un’interruzione sulla linea a cui è connesso. Ciascun dispositivo 1to4intgen consente, pertanto, mediante i suoi 4 timer, di generare 4 eventi autonomi ciclici e due eventi a programma.

Il dispositivo PIC acquisisce 8 linee di interruzioni dai due 1to4intgen e li invia codificate su 3 bit (IPL0, IPL1, IPL2) al processore.

Il codice assembler inizializza i due 1to4intgen e il PIC e assegna i vettori di interruzioni. Definisce le 8 ISR che dovranno gestire le 8 interruzioni inviate dai generatori.

Le ISR sono identiche e si limitano a stampare al video di TERM l’identificativo dell’interruzione ricevuta e gestita.

8 Il dispositivo DMA

Name	I8237DMA
Type	Ident. assoluto nella configurazione;
Address1	Indirizzo base per il dispositivo
Address2	Indirizzo base+$F
BUS	Ident. del bus a cui il dispositivo è connesso
COM1	Ident. della CPU master per il trasferimento
COM2	struttura interruzione (|vect|Priorità|linea|)
COM3	Ident. Gestore interruzioni
COM4	bit0-3 e bit4-7 ident. dispositivi connessi al can.0 e 1

Tabella 8

Le operazioni di trasferimento dati verso i moduli di I/O gestite mediante interruzioni o polling richiedono l’esecuzione di codici operativi da parte del processore per cui la velocità di trasferimento è limitata dalla velocità con cui il processore testa e serve il device e il processore è impegnato nell’eseguire un certo numero di istruzioni per ogni operazione di I/O. Quando grandi volumi di dati devono essere trasferiti, si riccorre ad un meccanismo più efficiente, quello basato sul DMA (Direct Memory Access) in grado di operare parallelamente al processore e gestire, mediante hardware specializzato le operazioni di trasferimento. Un DMA controller è, infatti, capace di diventare il master del bus e supervisionare un trasferimento tra la memoria ed un'interfaccia periferica o una memoria di massa senza intervento del processore. Mentre esegue un trasferimento, il DMA pone gli indirizzi di memoria sul bus e spedisce e riceve i segnali necessari per l'effettuazione di operazioni di lettura e scrittura in memoria e sulle interfacce periferiche.

Lo scopo di un DMA controller è quindi quello di realizzare una sequenza di trasferimenti mediante l'uso di cicli di bus sottratti al processore.

8.1 Generalità

Come dispositivo commerciale da cui derivare il DMA didattico inserito nell'ambiente ASIM, si è scelto l'Intel 8237, un controllore programmabile per l'accesso diretto in memoria (DMA) a 4 canali. Ciascun canale è una porzione del componente che serve una singola interfaccia. Le richieste in arrivo sui vari canali attraverso sono gestite secondo logica prioritaria fissa o rotante.

Il trasferimento dei dati può avvenire in 4 modi diversi:

- nel modo **single** il controllore dopo ogni trasferimento rilascerà il bus al processore per almeno un ciclo di bus, dopo inizierà di nuovo a testare la linea di richiesta e se attiva, procederà a "rubare" un altro ciclo;

- nel modo **block** la linea di richiesta è sufficiente che sia attiva solo fino al riconoscimento, dopo il quale il bus non sarà rilasciato fino al trasferimento dell'intero blocco;

- il modo **demand** è simile a quello **block** con la differenza che il trasferimento del blocco continua fin quando la linea di richiesta è attiva, ma, quando il trasferimento viene sospeso e poi ripreso esso inizia dal punto in cui era stato sospeso;

- il modo **cascade** permette di realizzare, collegando più controllori 8237 in cascata, sistemi DMA con più di 4 canali.

Il blocco di dati che è possibile trasferire in un'unica operazione è di 65536 byte. Alla fine del trasferimento viene avvisato il processore o la periferica che lo aveva richiesto attraverso una linea di fine conteggio posseduta dal controllore.

Altre caratteristiche del dispositivo sono: autoinizializzazione; trasferimenti da memoria a memoria; richiesta di DMA programmata; inibizione di un canale.

Il componente DMA simulato in ASIM differisce da quello commerciale per il minor numero di canali, che sono stati ridotti da 4 a 2 ed i modi di trasferimento che sono **single** e **block**. In fig.21 è mostrato il modello di programmazione del DMA con i registri e le linee fisiche simulate.

8.2 Configurazione

I dispositivi a cui si collega un DMA. controller in una configurazione sono:

- un processore, che avendo accesso ai registri del dispositivo, attraverso il sistema bus, permette la programmazione del componente;

- le periferiche alle quali vogliamo permettere l'accesso diretto in memoria.

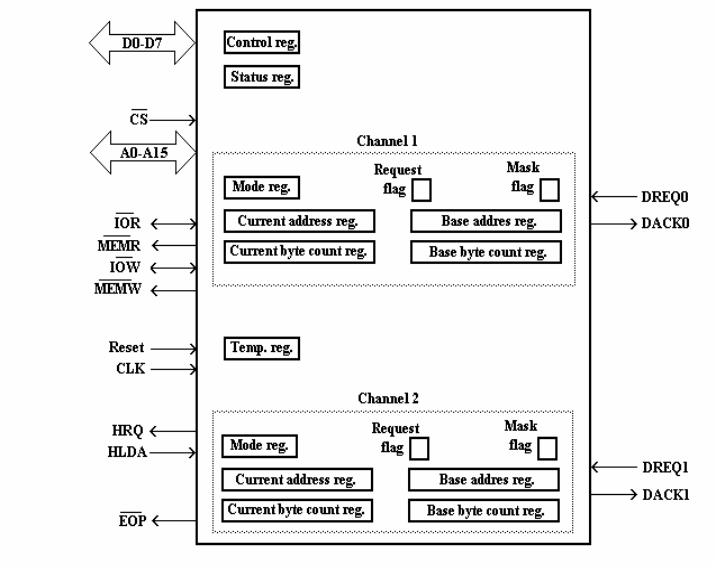


Fig. 21

8.3 Collegamento al processore

Le linee di collegamento al processore, sono mostrate sulla sinistra dello schema, in Fig. .

D0-D7 sono 8 linee che vanno connesse al bus dati per il trasferimento dei dati da e verso il componente.

La linea **CS** è utilizzata per la selezione del dispositivo, invece, i registri interni sono selezionati dai 4 bit meno significativi del bus indirizzo: **A0-A3**, e dai segnali di lettura-scrittura sul componente: **IOW** e **IOR**.

Come acccennato in precedenza, il controllore per eseguire un trasferimento deve spedire i segnali necessari per l'effettuazione di operazioni di lettura e scrittura in memoria (**MEMR** e **MEMW**) e sulle interfacce periferiche (**IOR** e **IOW**).

Sulle linee **CLK** e **Reset** arrivano rispettivamente il segnale dal generatore di clock e il segnale per cancellare tutti i registri del controllore.

La linea **HRQ** è adoperata per spedire una richiesta di controllo del sistema bus. Essa normalmente è applicata all'ingresso HOLD della CPU.

Invece, un segnale, in arrivo dalla CPU, sulla linea **HLDA** indica che è stato acquisito il sistema bus.

Infine la linea d'interruzione **EOP** trasmette, al processore o ad un eventuale gestore delle interruzioni, un interruzione per avvisare che il trasferimento di un blocco di memoria è stato completato.

Un esempio di collegamento del controllore ad un processore è mostrato in Fig. 20.

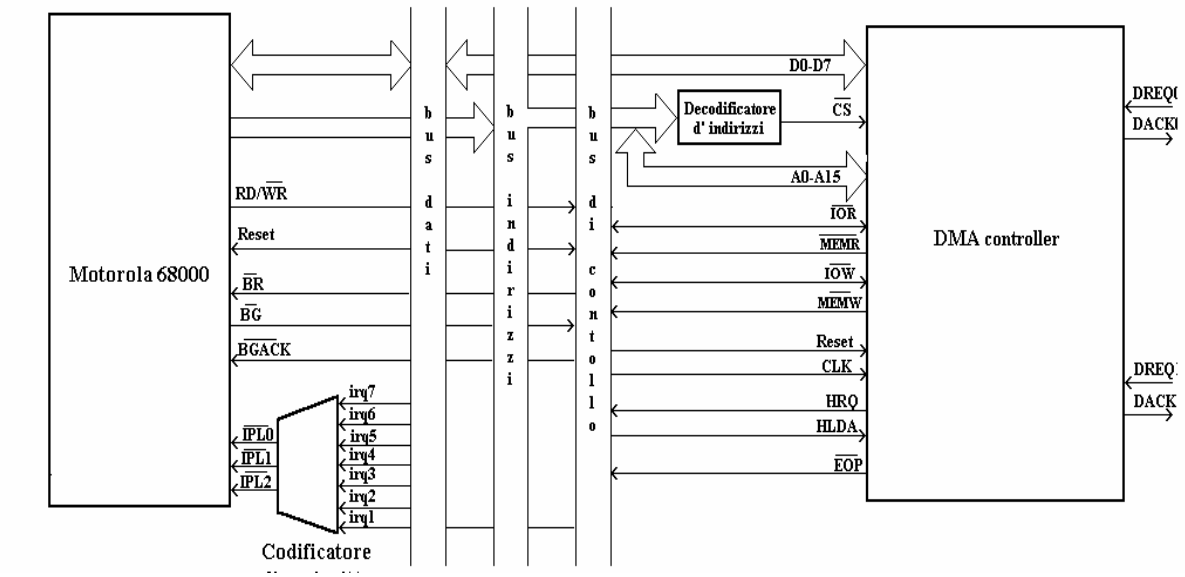


Fig. 20

8.4 Collegamento ad un device

Sulla destra dello schema in fig.1 sono mostrate le linee fisiche che collegano il DMA controller a delle periferiche.

DREQ0 e **DREQ1** sono le linee di richieste, usate dalle periferiche collegate ai rispettivi canali, per ottenere dei cicli DMA. Le richieste che arrivano su **DREQ0** hanno precedenza su quelle che arrivano su **DREQ1**.

DACK1 e **DACK2** informano la periferica, connessa a quel canale, che è stata selezionata per un ciclo DMA. Queste linee si comportano, nei confronti dei componenti periferici che richiedono questo servizio, come un "chip select".

Un esempio di connessione di due interfacce seriali è rappresentato in Fig. 21.

8.5 Collegamento software

Per inserire questo componente in una configurazione, bisogna attenersi alla stessa procedura seguita per gli altri componenti di tipo **Device**. I parametri presenti nella finestra **Aggiungi Device** permettono di gestire la connessione del controllore con gli altri componenti della configurazione. Il significato dei singoli parametri è il seguente:

Nome Elemento è utilizzato per specificare il tipo di componente da inserire, nel nostro caso deve essere I8237DMA.

Identificatore deve essere un numero compreso tra 01 ed FF e viene utilizzato dal programma per riferire il dispositivo.

Indirizzo1 rappresenta l'indirizzo più basso per indirizzare il componente, nel caso del controllore per l'accesso diretto in memoria esso deve essere un numero divisibile per 16.

Indirizzo2 definisce l'indirizzo più alto per indirizzare il componente, esso deve essere uguale ad Indirizzo 1 + F.

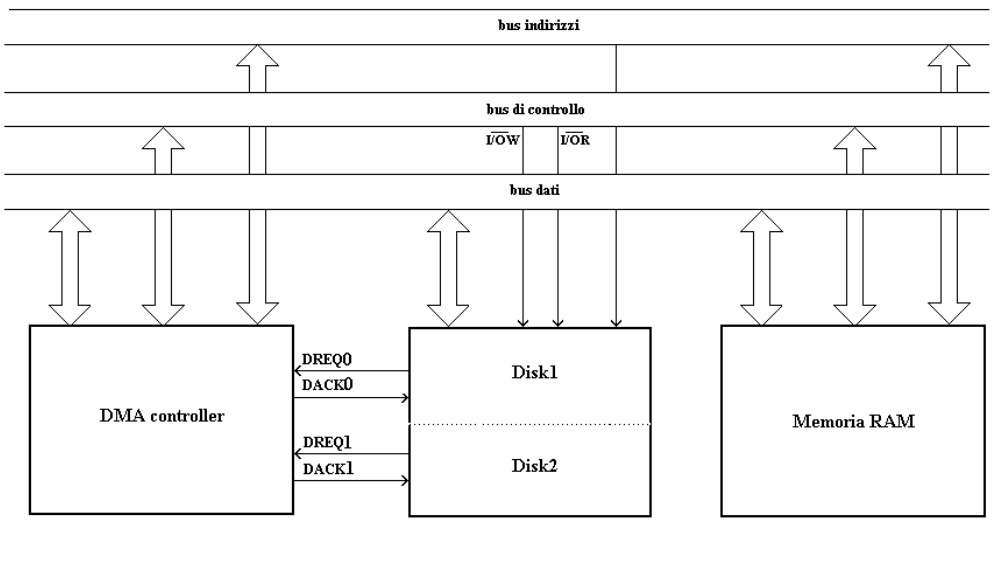


Fig. 21

Questi ultimi due parametri permettono di definire per quali indirizzi il decodificatore d'indirizzi attiva la linea **CS.** Infatti, se l'indirizzo sul bus indirizzo è compreso tra **Indirizzo1** e **Indirizzo2** , il componente viene selezionato, cioè l'operazione di lettura o scrittura è eseguita su un suo registro.

BUS determina l'Identificatore del bus a cui è connesso il dispositivo. Quindi, scegliendo per **BUS** l'identificatore di un componente MMU/BUS si connettono le linee **D0-D7**, **A0-A7**, **IOR**, **IOW**, **MEMR**, **MEMW**, **CLK** e **Reset** del DMA a quel sistema bus.

COM1 definisce l'Identificatore della CPU alla quale richiedere il controllo del bus e quindi a quale processore collegare le linee **HRQ** e **HLDA**.

COM2: viene utilizzato per specificare la struttura dati associata all'interruzione trasmessa al componente, specificato in **COM3**, quando termina un conteggio in un canale. Delle quattro cifre esadecimali, che definiscono **COM2** la meno significativa individua la linea d'interruzione, la seconda definisce la priorità e le due più significative specificano il "vector number" da trasmettere al processore che gestisce l'interruzione. Se le interruzioni sono gestite da un PIC, queste due cifre non devono essere specificate.

COM3: specifica l'Identificatore del gestore delle interruzioni che può essere sia un componente di tipo CPU che di tipo PIC.

I due parametri **COM2** e **COM3** permettono, quindi, di gestire la connessione della linea d'interruzioni **EOP** al processore o al PIC

COM4 le 2 cifre meno significative e le 2 più significative specificano rispettivamente l'Identificatore del componente collegato al canale 0 e al canale 1.

Specificando il valore in **COM4** si definisce a quali componenti collegare le coppie di linee (**DREQ0**, **DACK0**) e (**DREQ1**, **DACK1**).

8.6 Modello di programmazione

La finestra di programmazione associata al modulo I8237DMA ed i registri programmabili del controllore sono riportati in Fig. 22.

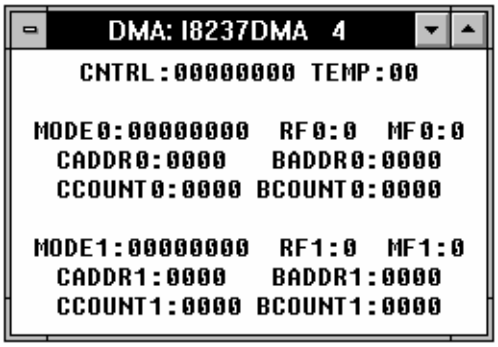


Fig. 22

Per accedere ad un registro occorre che esso sia compreso tra i valori **Indirizzo1** ed **Indirizzo2** (valori inseriti prima della creazione del dispositivo). La selezione dei registri nel DMA controller avviene attraverso i 4 bit meno significativi dell'indirizzo; questi formano l'**indirizzo relativo** con **valore che va da 0 ad F** esadecimale.

Nel dispositivo ai registri appartenenti al canale 0 è stato dato un nome terminante con "0" mentre a quelli appartenenti al canale 1 un nome terminante con "1".

I registri **CADDR0** e **CADDR1** di indirizzo corrente hanno parallelismo 16 bit e hanno il compito di contenere l'indirizzo della prossima locazione di memoria che deve partecipare al trasferimento. Nel caso del trasferimento da memoria a memoria **CADDR0** contiene l'indirizzo sorgente e **CADDR1** l'indirizzo destinazione. In ogni caso, entrambi i registri sono accessibili sia in lettura che in scrittura ed il loro indirizzo relativo è 0 per **CADDR0** ed è 2 per **CADDR1**.

BADDR0 e **BADDR1** sono i registri indirizzo di base ed hanno la funzione di conservare gli indirizzi iniziali rispettivamente di **CADDR0** e **CADDR1**. Essi sono a 16 bit e sono accessibili solo in fase di scrittura, infatti, quando viene scritto un valore in un registro indirizzo corrente, questo viene copiato anche nel relativo registro di base ed il valore rimane immutato fino a quando non si verifica un'altra scrittura.

I registri **CCOUNT0** e **CCOUNT1**, detti registri di conteggio correnti, sono anch'essi dei registri a 16 bit e mantengono il numero di byte che devono essere ancora trasferiti, sono accessibili in lettura e scrittura ed il loro indirizzo relativo è 1 per **CCOUNT0** e 3 per **CCOUNT1**.Nel caso di trasferimento da memoria a memoria il conteggio viene effettuato da **CCOUNT1**.

BCOUNT0 e **BCOUNT1** sono i registri a 16 bit detti di conteggio di base ed hanno la funzione di conservare gli indirizzi iniziali rispettivamente di **CCOUNT0** e **CCOUNT1**, essi sono accessibili solo in fase di scrittura. Quando viene scritto un valore in un registro di conteggio corrente questo viene copiato anche nel relativo registro di base ed il valore rimane immutato fino a quando non si verifica un'altra scrittura.

I registri, che contengono le informazioni relative al modo di funzionamento dei rispettivi canali, sono stati chiamati **MODE0** e **MODE1**. Essi possono essere accessi solo in scrittura ed hanno entrambi indirizzo relativo pari a B; la selezione tra i due avviene sul valore del bit meno significativo del dato: se 0, il dato viene scritto in **MODE0** altrimenti in **MODE1**. Il significato dei bit dei registri **MODE0** ed **MODE1** sono illustrati nella Errore. L'origine riferimento non è stata trovata**..**

RF0 e **RF1** rappresentano i flag delle richieste, cioè i flag dove indirizzare richieste di tipo software al DMA, queste producono gli stessi effetti di quelle provenienti dalle interfacce dei dispositivi. Questi flag sono accessibili solo in scrittura e l'indirizzo relativo per entrambi è 9. Anche in questo caso la selezione del canale avviene sul bit meno significativo del dato: 0 per il canale 0 ed 1 per il canale 1. Il valore che deve assumere il flag deve essere posto sul bit numero 3 del dato.

bit	significato
0	instrada il dato su un **canale**, è uguale a 0 per il canale 0 ed 1 per il canale 1
1/2	non utilizzati
3	indica la **direzione di trasferimento**: 0 per trasferimenti da memoria ad interfaccia, 1 da interfaccia a memoria
4	il valore 1 abilita l'**autoinizializzazione**, cioè al termine del conteggio i registri indirizzo e di conteggio correnti sono caricati con i valori dei rispettivi registri di base
5	il valore 1 abilita il **decremento** di una unità del valore contenuto in **CADDR** di quel canale dopo ogni trasferimento di un byte; deve essere posto a 0, se vogliamo l'**incremento**
6	non utilizzato
7	determina il **modo del trasferimento**: poniamo il valore 0 per il modo **Single** ed 1 per il modo **Block**; nel primo caso il bus viene rilasciato al processore alla fine di ogni trasferimento, viceversa, nel modo **block** il bus viene rilasciato dopo il trasferimento dell'intero blocco

Tabella 9-Significato dei bit nei registri MODE

Alri flag presenti nel nostro componente sono **MF0** e **MF1**. Questi mascherano le richieste dei rispettivi canali, cioè una richiesta non viene inoltrata al processore se il mask flag di quel canale è posto ad 1. **MF0** ed **MF1** sono accessibili solo in scrittura, per entrambi l'indirizzo relativo è A, il canale selezionato è pari al valore del bit 0 del dato ed infine il valore da inserire nel flag deve esssere posto sul bit 2 del dato.

bit	*azione svolta se il bit è posto ad 1*
0	**termine conteggio per il canale 0**
1	**termine conteggio per il canale 1**
2	è stata inoltrata una **richiesta al canale 0**
3	è stata inoltrata una **richiesta al canale 1**
4	non utilizzato
5	abilita il **trasferimento da memoria a memoria**
6	impone che in un trasferimento da memoria a memoria l'**indirizzo sorgente** deve rimanere **costante** per trasferire un byte in più locazioni di memoria
7	**abilita il DMA controller**

Tabella 10-Significato dei bit di CNTRL

In un trasferimento da memoria a memoria il byte da spostare viene letto dalla locazione sorgente e spostato in un registro temporaneo, che è stato chiamato **TEMP**, a questo punto viene realizzata una scrittura col valore di **TEMP** nella locazione di memoria di destinazione. Questo registro può essere solo letto ed ha indirizzo relativo pari a D.

L'ultimo registro da descrivere è **CNTRL**; esso è diviso in 2 parti: i 4 bit meno significativi rappresentano i bit di stato del componente, mentre quelli più significativi i bit di controllo. Su questo registro possono essere fatti accessi, all'indirizzo relativo 8, sia in lettura che in scrittura, queste ultime però non influenzano i 4 bit di stato. Il significato dei bit di **CNTRL** é specificato in .

Un quadro riassuntivo sull'indirizzamento dei registri e dei flag del componente è riportato in Tabella 11.

Indirizzo relativo esadecimale	*Indirizzo relativo binario*	*Tipo di accesso*	*Nome del registro o del flag*
0	0000	W/R	**CADDR0**
1	0001	W/R	**CCOUNT0**
2	0010	W/R	**CADDR1**
3	0011	W/R	**CCOUNT1**
8	1000	W/R	**CNTRL**
9	1001	W	**RF0/RF1**
A	1010	W	**MF0/MF1**
B	1011	W	**MODE0/MODE1**
D	1101	R	**TEMP**

Tabella 11-Indirizzamento dei registri e dei flag del DMA controller

8.7 Descrizione dei comandi

I comandi disponibili sono:

- **RESET** che riporta il componente nello stato iniziale azzerando tutti i registri; per attivarlo basta accedere in scrittura all'indirizzo relativo D;
- Clear Mask Flag (**CMF**) che cancella tutti i flag MF, si attiva scrivendo all'indirizzo relativo E;
- Write All Mask Flag (**WAMF**) che permette di fissare contemporaneamente tutti i flag **MF**; la scrittura deve avvenire all'indirizzo relativo F ed il valore di **MF0** deve essere posto nel bit 0 del dato, mentre il valore di **MF1** nel bit 1.

Indirizzo relativo esadecimale	Indirizzo relativo binario	Tipo di accesso	Nome del comando
D	1101	W	**RESET**
E	1110	W	**CMF**
F	1111	W	**WAMF**

Tabella 12-Indirizzi dei comandi del DMA controller

In Tabella 12 sono presentati gli indirizzi di attivazione dei comandi.

8.8 Programmazione

Prima che sia effettuata la richiesta di un trasferimento dati tra la memoria ed una interfaccia periferica si devono fissare i valori nei registri del canale interessato al trasferimento e nel registro di controllo.

Nell'ipotesi che il canale adoperato sia quello '0', le operazioni da eseguire sono le seguenti:

- scrivere all'indirizzo relativo **0** (registro **CADDR0**) una word indicante l'indirizzo del primo byte, del blocco in memoria, da trasferire;

- scrivere all'indirizzo relativo **1** (registro **CCOUNT0**) una word indicante il numero di byte che si vuole trasferire;

- scrivere all'indirizzo relativo **B** esadecimale (registro **MODE0**) un byte che indichi il modo di funzionare del canale (direzione del trasferimento, autoinizializzazione, incremento o decremento di **CADDR0**, modo del trasferimento **Single** o **Block**);

- scrivere all'indirizzo relativo **8** (registro **CNTRL**) un byte che abiliti il controllore.

Un primo esempio di inizializzazione è di seguito illustrato. In esso si presuppone che A0 sia stato precedentemente caricato con l'indirizzo più basso associato al DMA controller.

Queste istruzioni abilitano il controllore a realizzare un trasferimento di tipo SINGLE dalla memoria all'interfaccia collegata al canale 0. Il numero di byte da trasferire è 32 (20 esad.), gli indirizzi sono quelli crescenti da 1000 esad. in poi. Infine è stata abilitata l'autoinizializzazione, che permette di ricaricare, al termine del trasferimento, i registri **CADDR0** e **CCOUNT0** con i rispettivi registri di base, rendendo così il componente pronto ad eseguire un nuovo trasferimento.

```
MOVE.W#$1000,0(A0)
MOVE.B#$20,1(A0)
MOVE.B#$10,11(A0)
MOVE.B#$80,8(A0)
```

Il trasferimento inizia, comunque, quando la periferica collegata al canale '0'spedisce una richiesta al DMA e quest'ultimo l'accetta. La richiesta potrebbe anche realizzarsi via software con la seguente istruzione:

```
MOVE.B          #$08,9(A0)
```

Per un trasferimento di un blocco di byte all'interno della memoria le operazioni da eseguire sono le seguenti:

- scrivere all'indirizzo relativo **0** (registro **CADDR0**) e **2** (registro **CADDR1**) due word indicanti gli indirizzi del primo byte, del blocco in memoria, rispettivamente sorgente e destinazione;

- scrivere all'indirizzo relativo **3** (registro **CCOUNT1**) una word indicante il numero di byte che si vuole trasferire;

- scrivere due volte all'indirizzo relativo **B** esadecimale (registri **MODE0** e **MODE1**) un byte che indichi il modo di funzionare del canale 0 (sorgente) e del canale 1 (destinazione) per quanto riguarda l'autoinizializzazione e l'incremento o il decremento di **CADDR0** e **CADDR1**);

- scrivere all'indirizzo relativo **8** (registro **CNTRL**) un byte che abiliti il controllore ed il trasferimento da memoria a memoria.

Un esempio di inizializzazione su questo tipo trasferimento è riportato di seguito; anche in questo esempio in A0 è presente l'indirizzo più basso associato al DMA controller.

Queste istruzioni abilitano il controllore a realizzare un trasferimento in memoria di un blocco di 64 byte (40 esad.) a partire dall'indirizzo (sorgente) 1000 esad. in poi. Esso deve essere trasferito nella zona di memoria successiva all'indirizzo 2000 esad. (destinazione). Infine è stata disabilitata l'autoinizializzazione sia per il canale sorgente che per quello destinazione.

```
MOVE.W      #$1000,0(A0)
MOVE.W      #$2000,2(A0)
MOVE.B      #$40,3(A0)
MOVE.B      #$00,11(A0)
MOVE.B      #$01,11(A0)
MOVE.B      #$A0,8(A0)
```

8.9 Il comportamento.

Un DMA controller è progettato per servire una o più interfacce, nel nostro caso le interfacce collegabili sono massimo 2. Questa scelta è dettata dall'esiguo numero di parametri che definiscono un Chip in ASIM. Una convenzione stabilita per le periferiche che si collegano a un canale di questo dispositivo è che i dati sono trasferiti riferendosi al suo indirizzo più basso, cioè al valore del parametro **Indirizzo 1** di quella periferica.

Il controllore quando richiede il bus ad un processore blocca le operazioni di quest'ultimo fino al momento del rilascio il bus.

In questa descrizione, quando ci riferiremo ad un registro senza indicare il numero finale del suo nome, s'intenderà il registro relativo al canale sul quale si sta realizzando il trasferimento.

Quando un device collegato ad un canale del D.M.A. manda una richiesta, e il flag **MF** di quel canale è uguale a zero, viene settato il bit corrispondente in **CNTRL**: il bit 2, se la richiesta è pervenuta sul canale 0, o il bit 3, se la richiesta sopraggiunge sul canale 1. Se poi il bit 7 del registro **CNTRL** (che abilita il dispositivo) è uguale a uno, il dispositivo spedisce al processore una richiesta di bus il quale nel nostro caso viene automaticamente concesso.

La direzione del trasferimento dipende dal valore del bit 3 del registro **MODE** di quel canale, se pari a zero, il trasferimento si realizza dalla memoria al device, altrimenti avviene nella direzione opposta. Se, invece, il bit 5 di **CNTRL** è uguale ad 1, il trasferimento si verifica da memoria a memoria.

Una volta impossessatosi del bus il controllore effettua, per un trasferimento da Device a memoria, una lettura all'indirizzo più basso associato al device e una scrittura del dato letto all'indirizzo di memoria presente nel registro **CADDR** di quel canale, ovviamente, nel caso di trasferimento da memoria a device, le operazioni sono invertite.

Comunque in entrambi i casi viene decrementato di una unità il registro **CCOUNT** del relativo canale mentre il registro **CADDR** viene incrementato o decrementato di una unità rispettivamente, se il bit 5 del registro **MODE** è pari a zero o ad uno.

Il bus viene rilasciato al processore dopo il trasferimento del singolo byte o dell'intero blocco se rispettivamente il valore del bit 7 di **MODE** è zero o uno.

Al termine del trasferimento di un blocco, cioè quando il valore **CCOUNT** diventa nullo,

- viene cancellato il flag **RF** (è il flag dove un processore può indirizzare le sue richieste ed hanno gli stessi effetti di quelle che arrivano dai device),
- è azzerato il bit 2 o 3 di **CNTRL** (bit delle richieste) se abbiamo utilizzato rispettivamente il canale zero o uno,
- è posto ad uno il bit 0 o 1 di **CNTRL** (bit di fine conteggio) rispettivamente per il canale zero e uno,

- infine viene inviata un'interruzione del tipo specificato in **COM2** al gestore delle interruzioni specificato in **COM3**.

Per un trasferimento da memoria a memoria viene effettuata, prima una operazione di lettura all'indirizzo contenuto in **CADDR0** ed il dato viene posto nel registro **TEMP**, poi questo valore viene scritto all'indirizzo di memoria contenuto in **CADDR1**.

Dopo un trasferimento il registro **CCOUNT1** viene decrementato di una unità mentre **CADDR0** e **CADDR1** vengono incrementati o decrementati di una unità, se il bit 5 dei rispettivi registri **MODE** è pari a zero o ad uno. Se però il bit 6 di **CNTRL** è posto ad 1 allora **CADDR0** rimane costante durante il trasferimento.

Il bus viene rilasciato al processore dopo il trasferimento dell'intero blocco, cioè quando il valore in **CCOUNT1** diventa nullo. Alla fine del trasferimento viene azzerato il bit 5 di **CNTRL**, il quale attiva i trasferimenti da memoria a memoria.

Qualunque sia il tipo di trasferimento, se il bit 4 di un registro **MODE** è fissato ad 1, allora al termine del trasferimento di un blocco i registri **CADDR** e **CCOUNT** di quel canale sono caricati con i valori rispettivamente di **BADDR** e **BCOUNT**; questi registri in fase di scrittura di **CADDR** e **CCOUNT** assumono il loro stesso valore.

9 Il dispositivo PIA Periferal Interface Adapter

I campi relativi alla configurazione del dispositivo sono così codificati:

Name	MM6821PIA
Type	Identificatore assoluto nella configurazione
Address1	Indirizzo base
Address2	Indirizzo base+3
BUS	Identificatore di bus esterno a cui rendere visibile le memorie di bus/mem
COM1	Identificatore dispositivo gestore delle interruzioni generate dall'oggetto
COM2	Controllo linea Interruzione IRQA: XXYZ (XX=#vettore, Y=livello di priorita', Z=linea di interruzione
COM3	Controllo linea Interruzione IRQB: XXYZ (XX=#vettore, Y livello di priorità, Z=linea di interruzione
COM4	I due caratteri meno significativi contengono l'identificatore assoluto dell'oggetto a cui il porto parallelo e'connesso, i due caratteri più significativi la tipologia di connessione.

Tabella 13

Il dispositivo PIA (Periferal Interface Adapter) è un dispositivo parallelo a parallelismo 8 bit facente parte della famiglia dei processori Motorola. Il dispositivo contiene due sezioni quasi identiche, ciascuna dotata di 8 bit dati configurabili, anche singolarmente, come linee di ingresso o di uscita. Ciascun porto può, quindi, funzionare in ingresso, uscita o in configurazione mista (in-out). Due coppie di linee di controllo sono utilizzate per gestire l'handshaking per la sincronizzazione della periferica connessa al dispositivo. Di queste, CA1 e CB1 sono sempre di ingresso al dispositivo, e servono a controllare la linea di interruzione IRQA (o IRQB) verso il processore; le altre, CA2 e CB2, possono essere programmate per operare sia in ingresso (in tal caso si comportano in modo analogo alla CA1 e CB1) sia in uscita per implementare alcuni tipi di handshaking. Il controllo delle operazioni realizzabile con le due linee è programmato, come di seguito mostrato, mediante il vari campi del registro di controllo.

9.1 Programmazione del dispositivo PIA

Il dispositivo PIA simulato in ASIM è derivato da quello commerciale MC6821 di cui ne conserva, quasi tutte le caratteristiche funzionali e architetturali. In particolare la struttura dei sei registri interni ad otto bit:

- due registri per il trasferimento dei dati da e verso la periferica (PRA e PRB);
- due registri di controllo/stato (CRA e CRB);
- due registri, DRA e DRB, per il controllo della direzione dei dati (in input o in output).

Poiché al dispositivo sono riservati solo quattro indirizzi, per accedere a tali registri viene utilizzato un meccanismo di indirizzamento indiretto per tramite di campi del registro controllo.

Le due sezioni del dispositivo PIA sono programmabili per

Controllo delle linee CA1 e CB1: CA1 consente di controllare il flag di interruzione, IRQA1, presente nella parola di stato-controllo in posizione b7. In particolare, tale bit va alto a seguito della transizione attiva di CA1 (la transizione attiva è programmabile mediante il bit b1. In particolare è 1-0 se il bit b1=0, 0-1 se b1=1). Lo stato del flag IRQA1 determina quello della linea fisica di interruzione IRQA (connessa ad un processore) che genera una

interruzione se effettua la transizione 1-0. L'interruzione può essere mascherata mediante il bit b0 (abilitata se b0=1, mascherata se b0=0). Il flag b7 è automaticamente resettato dalla lettura del registro dato.

Quanto detto per CA1 vale anche per la linea CB1.

AD1	AD0	CRA2	CRB2	Registro Selezionato
0	0	1	X	PRA
0	0	0	X	DRA
0	1	X	X	CRA
1	0	X	1	PRB
1	0	X	0	DRB
1	1	X	X	CRB

X : indifferente.

Tabella 14-Indirizzamento dei registri interni

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	Controllo CA2			Accesso DRA	Controllo CA1	
	7	6	5	4	3	2	1	0
CRB	IRQB1	IRQB2	Controllo CB2			Accesso DRB	Controllo CB1	

Tabella 15-Formato del byte di controllo

Controllo delle linee CA2 e CB2: il controllo di CA2 è differente a seconda che tale linea sia stata programmata per operare come linea di ingresso o di uscita.

CA2 (o CB2) come linea di ingresso (b5=0): si comporta come CA1 con la differenza che il flag di interruzione interessato è IRQA2 ed i bit di programmazione sono b3 nelle funzioni di b0 e b4 nelle funzioni di b1. Quanto detto vale anche per CB2.

CA2 o (CB2) come linea di uscita (b5=1): in tal caso CA2 consente di controllare la periferica. Sono previsti 3 possibili modi di sincronizzazione codificati con i bit b4 e b3:

La , dove AD1 e AD0 sono i due bit meno significativi dell'indirizzo, mostra come selezionare i singoli registri. Ad esempio, PRA e DRA sono mappati allo stesso indirizzo, ma è possibile selezionarne uno dei due fissando prima il valore del bit 2 di CRA. Poiché il dispositivo ha un bus ad otto bit, le sole operazioni di lettura o scrittura consentite sono quelle sul byte.

CRA1 (CRB1)	CRA0 (CRB0)	Flag Interruzione CRA7 (CRB7)	Richiesta Interr IRQA (IRQB)
0	0	Alto su high/low di CA1 (CB1)	Disabilitata
0	1	Alto su high/low di CA1 (CB1)	Inviata quando CRA7 (CRB7) diventa alto
1	0	Alto su low/high di CA1 (CB1)	Disabilitata
1	1	Alto su low/high di CA1 (CB1)	Inviata quando CRA7 (CRB7) diventa alto

Il flag di interruzione CRA7 (CRB7) torna al valore basso in seguito ad un' operazione di lettura su PRA (PRB).

Tabella 16-Controllo della linea CA1(CB1)

I registri PRA, DRA, CRA consentono il controllo completo di un insieme di otto linee dato da connettere verso una periferica (linee A0..A7). Ciascuna linea può essere programmata separatamente come linea di uscita o di ingresso settando, rispettivamente ad 1 o a 0, il corrispondente bit del registro DRA. Quando si scrive su PRA, il valore dei singoli bit di PRA compare sulle corrispondenti linee dati programmate come output; quando si legge su PRA, i singoli bit di PRA assumono i valori presenti sulle linee programmate come input. Per gestire la sincronizzazione con la periferica, sono previste due ulteriori linee: CA1 e CA2. La linea CA1 è di sincronizzazione in ingresso. Le funzioni possono essere programmate, come mostrato in , definendo i due bit meno significativi di CRA; CA2 è una linea di sincronizzazione che può essere in input o in output; la , la Tabella 20 e la Tabella 19mostrano come programmare tale linea.

A provocare l'acquisizione o l'invio di un dato verso la periferica sono, in generale, le transizioni low/high o high/low delle linee di sincronizzazione.

Il significato dei singoli bit di CRA è mostrato in Tabella 15; i bit 6 e 7 sono a sola lettura e rappresentano i flag di interruzione; se le interruzioni sono abilitate, quando uno dei due flag diviene alto viene inviata al gestore delle interruzione l'IRQA (si veda la successiva descrizione dei parametri).

CRA5 (CRB5)	CRA4 (CRB4)	CRA 3 (CRB 3)	Flag Interruzione CRA6 (CRB 6)	Richiesta Interr IRQA (IRQB)
0	0	0	Alto su high/low di CA2 (CB2)	Disabilitata
0	0	1	Alto su high/low di CA2 (CB2)	Inviata quando CRA6 (CRB 6) diventa alto
0	1	0	Alto su low/high di CA2 (CB2)	Disabilitata
0	1	1	Alto su low/high di CA2 (CB2)	Inviata quando CRA6 (CRB6) diventa alto

Il flag di interruzione CRA6 (CRB6) torna al valore basso in seguito ad un' operazione di lettura su PRA (PRB).

Tabella 17-Effetti delle variazioni sulla linea CA2 (CB2) in ingresso

Per il gruppo di registri PRB, DRB e CRB vale quanto detto per il gruppo precedente; le differenze sono nelle funzioni attribuite alle linee di sincronizzazione come mostrato nelle tabelle.

La finestra associata a M6821PIA ne mostra tutti i registri (); il valore di tali registri può essere modificato, oltre che da programma, utilizzando il comando Modifica Valore del menù Device. E' possibile ridirigere l'input e l'output verso file anziché verso una periferica con i comandi Input da File e Output su File del menù Device.

Tabella 5 : Controllo della linea di uscita CB2

CRB5	CRB4	CRB3	CB2 BASSO	CB2 ALTO
1	0	0	Basso in seguito ad un' operazione di scrittura su PRB	Alto quando CRB7 va ad 1 per una variazione h/l o l/h di CB1
1	0	1	Basso in seguito ad un' operazione di scrittura su PRB	Alto al primo colpo di clock successivo la scrittura su PRB
1	1	0	Basso quando CRB3 diviene basso in seguito ad un' operazione di scrittura su CRB	Sempre basso finche CRB3 è basso. Diviene alto se CRB3 va ad 1 per un' op. di scritt. su CRB
1	1	1	Sempre alto finchè CRB3 è alto Diviene basso se CRB3 va a 0 per un' op. di scrittura su CRB	Alto quando CRB3 diviene alto in seguito ad un' operazione di scrittura su CRB

Tabella 18

Tabella 6 : Controllo della linea di uscita CA2

CRA5	CRA4	CRA3	CA2 BASSO	CA2 ALTO
1	0	0	Basso in seguito ad un' operazione di lettura su PRA	Alto quando CRA7 va ad 1 per una variazione h/l o l/h di CA1
1	0	1	Basso in seguito ad un' operazione di lettura su PRA	Alto al primo colpo di clock successivo alla lettura su PRA
1	1	0	Basso quando CRA3 diviene basso in seguito ad un' operazione di scrittura su CRA	Sempre basso finche CRA3 è basso. Diviene alto se CRA3 va ad 1 per un' op. di scritt. su CRA
1	1	1	Sempre alto finchè CRA3 è alto Diviene basso se CRA3 va a 0 per un' op. di scrittura su CRA	Alto quando CRA3 diviene alto in seguito ad un' operazione di scrittura su CRA

Tabella 19

Per aggiungere un M6821PIA ad una configurazione si deve selezionare dal menù Configura il comando Aggiungi Device e specificare tutti i parametri previsti nella finestra di dialogo.

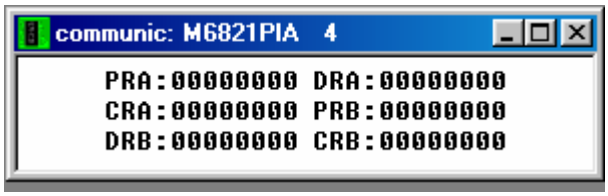


Fig. 23

Il "Nome Elemento" è M6821PIA. L'Indirizzo 1 deve essere pari ed "Indirizzo 2" deve essere uguale a "Indirizzo 1" + 3. "BUS" deve contenere l'Identificatore del bus cui il dispositivo è connesso. "Com1" contiene, se previsto, l'Identificatore del dispositivo gestore delle interruzioni. "Com2" e "Com3" definiscono le linee di interruzione IRQA e IRQB previste dal dispositivo (associate ai flag di interruzione, rispettivamente, dei registri CRA e CRB); i valori attribuibili a tali parametri sono quelli già più volte specificati; si osservi che, se "Com1" è diverso da zero, le linee di interruzione devono essere specificate e non possono essere nulle.

Infine, "Com4" viene utilizzato per specificare l'eventuale dispositivo periferico cui M6821PIA è connesso; nell'attuale versione di ASIM, l'unico dispositivo cui un M6821PIA può essere connesso è un altro M6821PIA. L'identificatore del dispositivo da connettere va nelle due cifre meno significative di "Com4". Le altre due cifre servono per definire le connessioni; di queste la meno significativa deve assumere uno dei seguenti valori:

- 0 linea CA2 connessa alla linea CA1 del dispositivo periferico;
- 1 linea CA2 connessa alla linea CA2 del dispositivo periferico;
- 2 linea CA2 connessa alla linea CB1 del dispositivo periferico;
- 3 linea CA2 connessa alla linea CB2 del dispositivo periferico;

nei primi due casi le linee dati A sono connesse alle linee dati A del dispositivo periferico; nei rimanenti due, le linee dati A sono connesse alle linee dati B del dispositivo periferico.

La più significativa deve assumere uno dei seguenti valori:

- 0 linea CB2 connessa alla linea CA1 del dispositivo periferico;
- 1 linea CB2 connessa alla linea CA2 del dispositivo periferico;
- 2 linea CB2 connessa alla linea CB1 del dispositivo periferico;
- 3 linea CB2 connessa alla linea CB2 del dispositivo periferico;

nei primi due casi le linee dati B sono connesse alle linee dati A del dispositivo periferico; nei rimanenti due, le linee dati B sono connesse alle linee dati B del dispositivo periferico.

E' cura di chi definisce la configurazione fare in modo che le connessioni definite per i due dispositivi siano coerenti. Inoltre, poiché all'accensione della macchina da simulare tutti i registri sono nulli, chi programma deve opportunamente inizializzare i dispositivi prima di tentare un trasferimento dati.

10 Il dispositivo video-tastiera TERMINAL

I campi relativi alla configurazione del dispositivo sono così codificati:

Name	TERMINAL
Type	Identificatore assoluto nella configurazione
Address1	Indirizzo registro dati (tastiera in scrittura, video in lettura)
Address2	Indirizzo registro stato-controllo
BUS	Identificatore di bus esterno a cui è connesso l'oggetto
COM1	Identificatore dispositivo gestore delle interruzioni generate dall'oggetto
COM2	Controllo Interruzione dovuta ad ENTER: XXYZ (XX=#vettore, Y=livello di priorità , Z=linea di interruzione
COM3	Idem a COM2 ma per interruzione dovuta a Buffer Full
COM4	n.s. va posto a 0

Tabella 20

ASIM mette a disposizione, come dispositivo base per realizzare l'I/O da e per un sistema, un dispositivo da utilizzare, appunto, come terminale video- tastiera denominato TERMINAL.

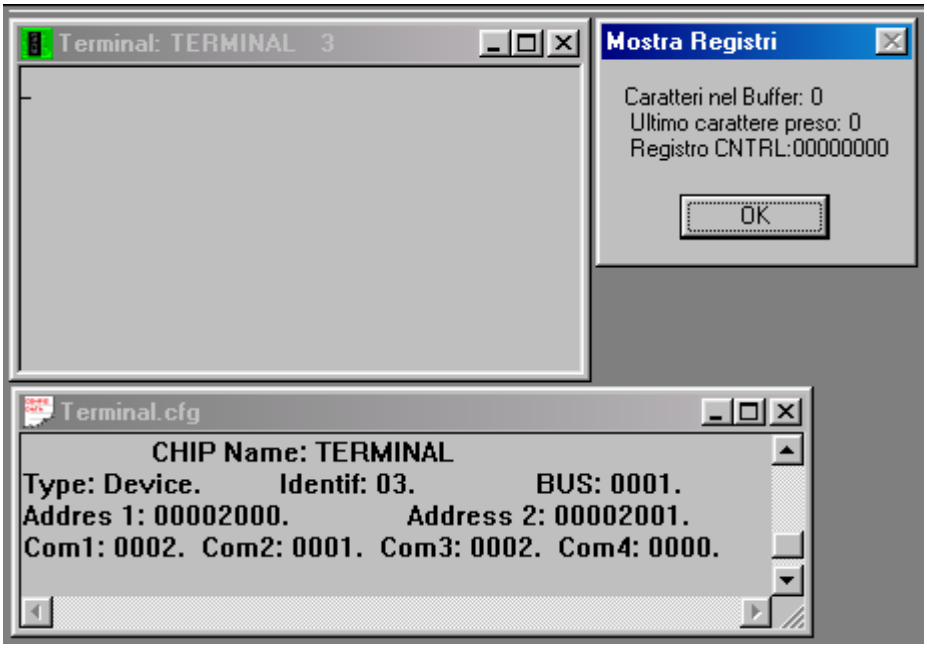


Fig. 24-Finestre ASIM associate al TERMINAL

Nella finestra associata a questo dispositivo (Fig. 24) appare l'output destinato al terminale compreso l'eco dell'input da tastiera. Il video simulato è alfanumerico con dieci righe di quaranta caratteri; la tastiera coincide con quella del PC IBM (eslusi i tasti speciali: Ctrl, Alt, Esc, tasti funzione e movimento cursore) e prevede i tasti di "backspace" ed "enter".

Quando la finestra di TERMINAL diviene quella corrente, la tastiera del PC (esclusi i tasti speciali che svolgono le solite funzioni previste da Windows) emula quella del dispositivo nel senso che un eventuale input da tastiera appare nella finestra e viene inviato al sistema che si sta simulando. Più precisamente, la tastiera

accetta dei caratteri solo se è abilitata e l'input va a video solo se la funzione di eco è attiva. Dopo l'accensione della macchina che si sta simulando sia la tastiera sia l'eco sono disabilitati; per attivarli occorre modificare il valore del registro di controllo e stato: CNTRL. Questo è un registro di un sol byte i cui bit, a partire dal meno significativo, svolgono, se posti al valore 1, le seguenti funzioni:

Bit di Controllo

- bit0 abilita interruzione su "Buffer full";
- bit1 abilita interruzione sull'"ENTER";
- bit2 cancella video;
- bit3 pulisci buffer di tastiera;
- bit4 abilita tastiera;
- bit5 abilita eco;

Bit di Stato

- bit6 stato di "Buffer full";
- bit7 stato di ENTER inviato.

Il contenuto del registro CNTRL può essere modificato o da programma o utilizzando il comando *Modifica Valore* del menù *Device.* Il comando *Mostra Registri* apre una finestra di dialogo in cui appare il registro CNTRL, il numero di caratteri presenti nel buffer di tastiera e la posizione nel buffer dell'ultimo carattere letto dal processore (Fig. 24).

Per comprendere il significato di questi due numeri è necessario conoscere il meccanismo di gestione dell'input e output da parte del dispositivo in esame.

TERMINAL ha due indirizzi d'accesso: il primo è associato al buffer di tastiera e al video, il secondo al registro CNTRL. Un'operazione di accesso in scrittura sul primo indirizzo comporta la stampa a video del carattere scritto; i caratteri validi sono quelli con codice ASCII compreso tra 32 e 126 (estremi inclusi) ed il carattere con codice 13 che provoca il ritorno a capo. Quando una riga è piena, il successivo carattere viene scritto nella riga seguente; se tutte le dieci righe sono piene il testo scorre di una riga verso l'alto. Ponendo ad 1 il bit 2 di CNTRL, si pulisce lo schermo e si riporta il cursore in alto a sinistra.

Un'operazione di accesso in lettura sul primo indirizzo consente di prelevare un carattere dal buffer di tastiera. Partendo dallo stato di buffer di tastiera vuoto e con tastiera ed eco abilitati, si digiti una frase qualsiasi. Per ogni carattere premuto, il corrispondente carattere va nel buffer di TERMINAL ed appare nella finestra; inoltre il numero di caratteri nel buffer si incrementa di un'unità. Se vengono battuti più di 256 caratteri la tastiera viene disabilitata, il bit 6 di CNTRL va ad 1 e, se il bit 0 è alto, viene inviata un'interruzione per segnalare al processore la condizione di "Buffer full". Quando si è completata la frase (con meno di 256 caratteri) va premuto il tasto di ENTER; ciò porta TERMINAL a disabilitare la tastiera, ad alzare il bit 7 di CNTRL ed ad inviare al processore un interruzione se il bit 1 di CNTRL è alto; ogni qualvolta il processore legge un carattere dal buffer, il numero di caratteri letti, inizialmente nullo, si incrementa di uno; tale numero rappresenta un puntatore al primo carattere non letto e, il fatto che si incrementi man mano, fa si che il processore legga, nel corretto ordine, tutti i caratteri nel buffer. L'ultimo carattere è sempre l'ENTER; per riattivare la tastiera e pulire il buffer occorre portare al valore 1 i bit 3 e 4 di CNTRL.

Ponendo a 0 il bit 5 di CNTRL si disabilita l'eco; ciò è utile quando si vuole nascondere l'input di tastiera come nel caso in cui venga digitata una password.

Per aggiungere un terminale ad una macchina da simulare va selezionato il comando *Aggiungi Device* del menù *Configura* e vanno specificati i parametri nell'apposita finestra di dialogo (figura 3). Il "Nome Elemento" è TERMINAL; l'"Indirizzo1" DEVE essere pari ed è quello associato al video e buffer di tastiera; l'"Indirizzo2" DEVE essere quello successivo ad "Indirizzo 1" ed è quello associato a CNTRL.

In "BUS" va posto l'Identificatore del bus cui è connesso il dispositivo. In "Com1" va posto, se previsto, l'Identificatore del gestore delle interruzioni. "Com2" e "Com3" vengono utilizzati per specificare le linee di interruzione, rispettivamente, per ENTER e "Buffer full"; delle quattro cifre esadecimali che definiscono questi parametri, la meno significativa individua la linea di interruzione, la seconda definisce la priorità e le due più significative specificano il "vector number". Se "Com1" è diverso da zero, la linea (cifra meno significativa di "Com2" e "Com3") DEVE essere diversa da zero; le altre ciftre possono essere nulle. Infine "Com4" DEVE essere lasciato al valore zero.

11 Il dispositivo USART

I campi relativi alla configurazione del dispositivo sono così codificati:

Name	I8251USART
Type	Id. assoluto nella configurazione (01->FF)
Address1	Indirizzo base device (deve essere pari)
Address2	Indirizzo base+1
BUS	Identificatore di bus esterno a cui è connesso il device
COM1	Identificatore dispositivo gestore delle interruzioni generate dall'oggetto
COM2	Controllo linea Interruzione Rx
COM3	Controllo linea Interruzione Tx
COM4	xxyz: xx=id device connesso; y=0 connessione completa; y=1 connessione semplice;z=n.c.

Tabella 21

Per inserire un'USART in una configurazione ASIM i parametri precedenti vanno così configurati:

Name: va posto ad 8251USART.

Identificatore deve essere un *numero compreso tra 01 ed FF* e viene utilizzato dal programma per riferirsi a questo dispositivo.

Indirizzo1 rappresenta l'indirizzo più basso per accedere al componente, in questo caso esso deve essere un *numero pari.*

Indirizzo2 definisce l'indirizzo più alto per indirizzare il componente, esso deve essere *uguale ad Indirizzo1 + 1.*

Questi ultimi due parametri permettono di definire per quali indirizzi il decodificatore d'indirizzi attiva la linea **CS**. Infatti, se l'indirizzo sul bus indirizzo è pari ad uno dei due, il componente viene selezionato, cioè l'operazione di lettura o scrittura è eseguita su un suo registro.

BUS determina l'*Identificatore del bus* a cui è connesso l'interfaccia seriale. Quindi, scegliendo per **COM1** l'identificatore di un componente MMU/BUS si connettono le linee **bus dati**, **A0**, **RD**,**WR** e **Reset**, dell'interfaccia seriale, al bus suddetto.

COM1 definisce l'*Identificatore del gestore delle interruzioni,* cioè il componente (di tipo CPU o I8259PIC) a cui trasmettere le interruzioni.

COM2 viene utilizzato per specificare l'interruzione trasmessa al gestore delle interruzioni, quando viene ricevuto un carattere in **Receiver shift register** ed è copiato in **Data-in buffer register**, per essere letto dal processore. Delle quattro cifre esadecimali che definiscono **COM2** la meno significativa individua la *linea d'interruzione*, la seconda definisce la *priorità* e le due più significative specificano il "*vector number*" da trasmettere al processore che gestisce l'interruzione. Se le interruzioni sono gestite da un PIC, queste due cifre non devono essere specificate.

COM3 specifica l'interruzione inviata al gestore delle interruzioni, quando viene copiato il carattere dal **Data-out buffer register** in **Transmitter shift register** ed inizia la sua trasmissione. Le quattro cifre esadecimali hanno lo stesso significato di quelle di **COM2**.

E' chiaro, quindi, che nel nostro simulatore, l'insieme dei parametri **COM1**, **COM2** e **COM3**, permettono di gestire la connessione delle linee d'interruzioni, **RxRDY** e **TxRDY**, con il processore o il PIC

COM4 viene impiegato per specificare il device a cui l'interfaccia seriale è connessa ed il tipo di connessione. Precisamente, le 2 cifre meno significative specificano l'Identificatore del Device a cui è connesso l'interfaccia seriale. La terza cifra può assumere il valore '0'per realizzare un collegamento pieno o '1'per uno parziale.

Quindi, **COM4** consente di simulare la connessione delle **linee di handshaking**, di **Tx** e **Rx**, con un componente connettibile all'interfaccia seriale, cioè, che soddisfa le proprietà 1 e 2 descritte nel paragrafo precedente.

11.1.1 Protocollo di comunicazione asincrono

11.1.2 Protocollo di comunicazione sincrono

11.1.3 Generalità USART

Il dispositivo USART (Universal Synchronous-asynchronous Receiver Transmitter) consente di effettuare il collegamento seriale di un sistema verso un dispositivo periferico per effettuare un trasferimento di dati secondo un protocollo sincrono o asincrono. Le caratteristiche principali dei protocolli di trasmissione sincroni e sincroni sono discusse di seguito. Ad esempio un USART permette di collegare una periferica seriale (ad esempio un terminale, un modem, una stampante, etc.) ad un calcolatore o interconnettere due calcolatori.

Il componente commerciale di riferimento simulare l'USART in ASIM è l'Intel 8251A. Il modello funzionale a blocchi dell'USART è riportato in fig.26a, quello di programmazione con le linee di I/O dell'UART simulato sono riportati in fig.26.

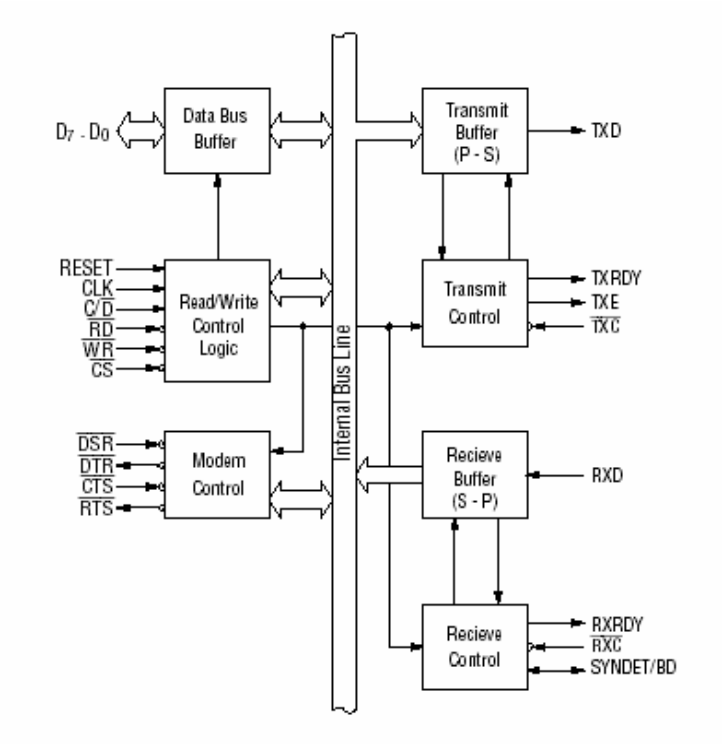


Fig.26a - Diagramma funzionale dell'USART I8251

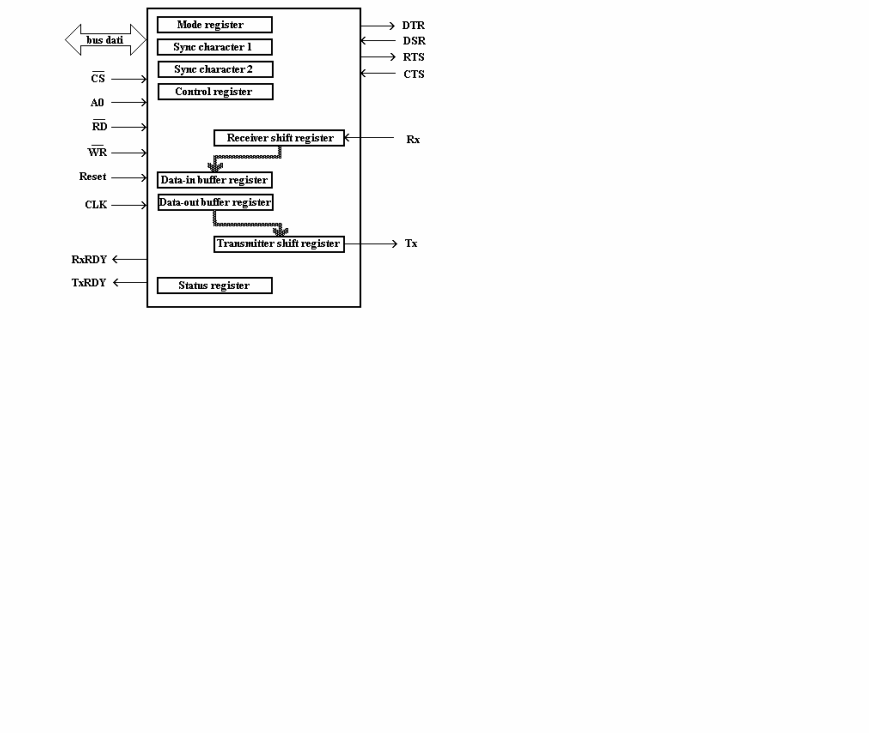


Fig.26b - Modello del dispositivo USART simulato in ASIM

Questo componente simula gli stessi registri e relative posizioni dei bit del componente commerciale. Per i dettagli tecnici è, pertanto, possibile riferire il data sheet dell'8251 riportati nell'apposito allegato.

L'USART è un dispositivo full duplex che accetta i caratteri dalla CPU in formato parallelo e poi li converte in un flusso continuo di dati seriali per trasmetterli in modo sincrono o asincrono. Simultaneamente, esso può ricevere un flusso di dati seriali e convertirli nel formato parallelo per essere letti dalla CPU.

Il componente segnala alla CPU quando esso è pronto ad accettare un nuovo carattere da trasmettere e quando ha ricevuto un carattere che può essere letto dalla CPU stessa; quest'ultima può leggere lo stato completo dell'interfaccia in qualsiasi momento.

Altre caratteristiche importanti sono:

- comunicazione di tipo sincrona o asincrona,
- trasmettitore e ricevitore full duplex con doppia bufferizzazione,
- caratteri da 5 ad 8 bit,
- inserzione automatica dei caratteri di sincronismo,
- rilevazione di errori di parità, di overrun e di framing.

11.1.4 Interfacci verso il BUS

Le linee fisiche, che collegano, attraverso il sistema bus, l'interfaccia seriale al processore, sono mostrate sulla sinistra dello schema, in Fig..

Il processore scambia i dati con il componente attraverso il **bus dati** e seleziona lo stesso mettendo sul bus indirizzi uno dei due indirizzi ad esso associati. Poi, un decodificatore di indirizzi, collegato al bus indirizzi, quando rivela uno degli indirizzi associati all'8259A, seleziona il componente attraverso la linea **CS**.

I registri interni sono selezionati dal bit meno significativo del bus indirizzo: **A0**; dai segnale di lettura-scrittura: **RD** e **WR**; oltre che dallo stato interno.

Un segnale sulla linea **Reset** riporta il componente nello stato iniziale cancellando tutti i suoi registri. Questo segnale può essere spedito sia dal processore che da un dispositivo esterno.

RxRDY e **TxRDY** sono due linee d'interruzione che trasmettono al processore o all'eventuale PICuna richiesta d'interruzione.

L'interruzione su **RxRDY** è inviata quando viene ricevuto un carattere in **Receiver shift register** ed è copiato in **Data-in buffer register**.

Su **TxRDY**, invece, l'interruzione viene inviata quando viene copiato il carattere dal **Data-out buffer** register in **Transmitter shift register** ed inizia la trasmissione.

Un esempio di collegamento dell'interfaccia seriale con un processore è mostrato in Fig. 25, dove sono presentate, del Motorola 68000, solo le linee coinvolte nel collegamento.

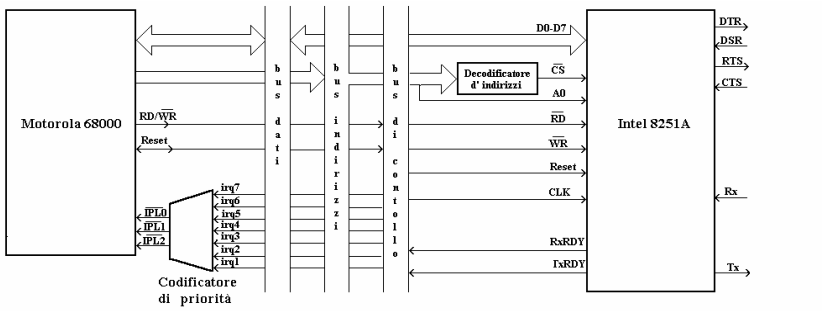


Fig. 25 Collegamento 8251A-MC68000

11.1.5 Interfaccia verso altri dispositivi

In generale due dispositivi sono collegabili quando hanno i segnali d'ingresso e uscita compatibili tra loro. Ciò significa che le uscite di un dispositivo devono essere dello stesso tipo delle entrate dell'altro e viceversa, inoltre, deve essere possibile rispettare i protocolli di handshaking di entrambi i dispositivi.

Sulla destra dello schema in figura 1 sono mostrate le linee fisiche che collegano l'interfaccia seriale ad un device. Le cinque linee poste più in alto vengono utilizzate per lo sviluppo del protocollo di handshaking ed il loro significato è fornito in .

Le altre due linee, **TX** e **RX**, sono utilizzate rispettivamente per trasmettere e ricevere i dati.

Le proprietà che deve possedere il componente che interfaccia la periferica seriale sono:

1) deve essere provvisto di linee compatibili alle seguenti 6 linee: **DTR**, **DSR**, **RTS**, **CTS**, **TX** ed **RX**, per una connessione piena con l'interfaccia seriale , altrimenti, se vogliamo realizzare una connessione parziale, deve disporre solo di linee compatibili a **TX** ed **RX**,

2) deve rispettare il protocollo di handshaking,

3) nel caso di comunicazione asincrona la trasmissione deve avere il formato presentato in Tabella 11.

segnale	significato
DTR	l'interfaccia chiede al modem di connettersi alla linea
DSR	il modem segnala all'interfaccia che si è connesso alla linea
RTS	l'interfaccia chiede al modem di trasmettere
CTS	il modem invia in linea la portante e segnala all'interfaccia che è pronto a trasmettere

Tabella 22-Segnali per lo sviluppo del protocollo di handshaking

In Fig. 27 sono mostrate le evoluzioni dei segnali di handshaking dell'interfaccia che trasmette i dati nel caso in cui dopo la trasmissione del messaggio non venga sconnesso il modem dalla linea.

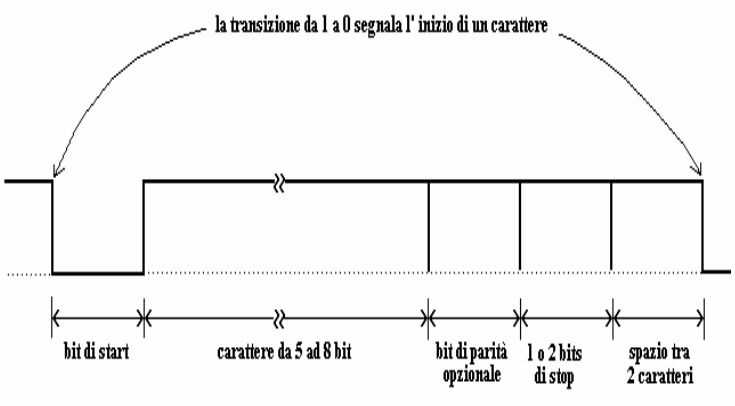


Fig. 26 Formato frame della trasmissione asincrona

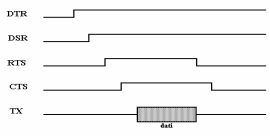


Fig. 27 Handshaking per la trasmissione Tx

Con riferimento all'esempio precedente sono mostrate in Fig. 28 le evoluzioni dei segnali di handshaking dell'interfaccia ricevente.

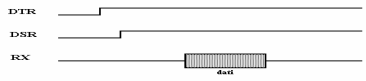


Fig. 28 Segnali di hanshaking dell'interfaccia Rx

Un esempio di connessione di due interfacce seriali è rappresentato in Fig. 29.

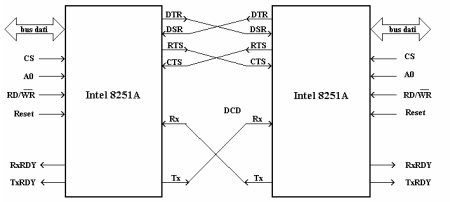


Fig. 29 Connessione completa fra dispositivi seriali

Oltre a questo tipo di connessione che citerò con l'aggettivo "piena" è possibile simulare anche una connessione "parziale", la quale coinvolge solo le linee **RX** e **TX** come è mostrato in Fig. 30.

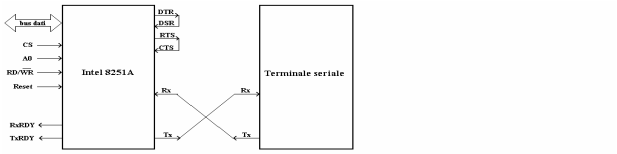


Fig. 30-Connessione parziale fra dispositivi seriali

11.1.6 Collegamento software

Fig. 31-La finestra dei registri del dispositivo USART

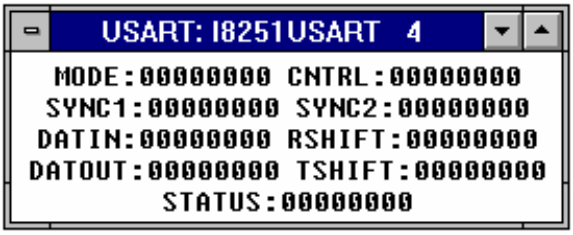
11.1.7 Modello di programmazione

I registri programmabili dell'interfaccia seriale sono quelli presenti in Fig. 31 con l'esclusione dei due shift register **TSHIFT** e **RSHIFT**.

Per accedere ad uno di questi 7 registri ad 8 bit occorre che l'indirizzo sia uguale a **Indirizzo1** o **Indirizzo2**. La selezione dei registri in questo dispositivo avviene attraverso il bit meno significativo dell'indirizzo, il tipo di accesso e lo stato del sistema.

Il registro che contiene le informazioni relative al genere di trasmissione che vogliamo realizzare è stato chiamato **MODE**. Esso può essere accesso solo in scrittura, all'indirizzo dispari e quando l'interfaccia è stata appena resettata. Il significato dei bit nel registro **MODE** sono illustrati nella Tabella 23.

Tabella 23-Significato dei bit nel registro MODE



Se con il valore inserito in **MODE** abbiamo deciso di realizzare una trasmissione sincrona, la prossima scrittura all'indirizzo dispari ci permetterà di inserire il primo carattere di sincronismo nel registro **SYNC1**, invece ,

bit	significato
0	determina il **tipo di trasmissione**, è fissato a 0 per ottenere una trasmissione sincrona e ad 1 per quella asincrona
1	non utilizzato
2/3	indicano il **numero di bit d'informazione** per ogni carattere trasmesso: 00 per 5 bit ,01 per 6, 10 per 7 e 11 per 8
4	il valore 1 di questo bit abilita la **presenza del bit di parità** nel carattere trasmesso
5	il **tipo di parità** è pari se il bit ha valore 1 altrimenti è dispari
6	determina in una trasmissione asincrona il **numero di bit di stop** per ogni carattere trasmesso: 0 per un solo bit di stop e 1 per 2
7	in una trasmissione sincrona definisce il **numero di caratteri di sincronismo** che devono essere trasmessi all'inizio della trasmissione: è posto a 0 per un carattere o ad 1 per due caratteri

qualora abbiamo deciso che i caratteri di sincronismo devono essere due, dobbiamo effettuare una ulteriore scrittura all'indirizzo dispari per inserire il secondo carattere di sincronismo in **SYNC2**.

Dopo la scrittura nel registro **MODE,** se trasmettiamo in modo asincrono, o dopo l'inserimento del o dei caratteri di sincronismo, se, invece, trasmettiamo in modo sincrono, ogni ulteriore accesso in scrittura all'indirizzo dispari viene eseguito nel registro **CNTRL**, il quale controlla il funzionamento dell'interfaccia seriale.

Il significato dei bit nel registro **CNTRL** sono illustrati nella Tabella 24.

Ovviamente, se resettiamo il componente, la sequenza di scrittura nei vari registri citati si ripete iniziando dal registro **MODE**.

Nel registro **DATIN** viene copiato il carattere che è stato ricevuto in **RSHIFT** per essere letto dal processore. **DATIN** è accessibile solo in lettura all'indirizzo pari. Un accesso in lettura in esso azzera il bit 1 di **STATUS**.

bit	azione svolta se il bit è posto ad 1
0	abilita il trasmettitore
1	attiva il segnale di handshaking DTR
2	abilita il ricevitore
3	non utilizzato
4	cancella i 3 bit d'errori nel registro STATUS
5	attiva il segnale di handshaking RTS
6	resetta l'interfaccia seriale
7	conduce il ricevitore nello stato "hunt", in questo stato il ricevitore cerca il o i caratteri di sincronismo memorizzati in SYNC1 e SYNC2

Tabella 24-Significato dei bit nel registro CNTRL

Nel registro **DATOUT** viene scritto dal processore il carattere che deve essere copiato in **TSHIFT** per essere trasmesso. **DATOUT** è accessibile solo in scrittura all'indirizzo pari; un tale tipo di accesso azzera il bit 0 di **STATUS**.

I registri **RSHIFT** e **TSHIFT** sono degli shift register non accessibili, ma vengono utilizzati dall'interfaccia per effettuare rispettivamente la trasformazione del formato serie/parallelo in ricezione e parallelo/serie in trasmissione.

Nel registro **STATUS** sono contenute informazioni sull'interfaccia che possono essere utilizzate dal programma che è in esecuzione; esso è accessibile in lettura all'indirizzo pari. Il significato dei bit nel registro **STATUS** sono illustrati nella Tabella 25.

Si riscontra un errore di parità, quando il numero di '1'nei bit d'informazione più il bit di parità non corrisponde al tipo di parità prevista.

bit	informazione indicata quando assume il valore 1
0	è stato copiato il carattere da DATOUT in TSHIFT ed inizia la trasmissione, questo bit si azzera quando il processore scrive un nuovo carattere in DATOUT
1	è stato ricevuto un carattere in RSHIFT ed è stato copiato in DATIN, questo bit si azzera quando il processore legge il dato da DATIN
2	in una trasmissione sincrona il trasmettitore non ha nessun carattere da trasmettere
3	è stato rilevato un errore di parità
4	è stato rilevato un errore di overrun
5	è stato rilevato un errore di framing
6	è stato rilevato il o i caratteri di sincronismo previsti
7	è stato attivato il segnale di handshaking DSR

Tabella 25-Significato dei bit nel registro STATUS

Un errore di overrun si verifica quando viene ricevuto un nuovo dato in DATIN, prima che il precedente sia stato letto dal processore, oppure quando il processore inserisce il prossimo carattere in DATOUT, prima che venga completamente trasmesso il dato precedente.

Un errore di framing si riscontra quando in una trasmissione asincrona il ricevitore si aspetta di rilevare un bit di stop e, invece, rileva uno zero.

Un quadro riassuntivo sugli indirizzamenti dei vari registri è mostrato in Tabella 26.

- il registro che viene selezionato dipende dallo stato del sistema.

Indirizzo	Tipo di accesso	Registro
Pari	R	**DATIN**
Pari	W	**DATOUT**
Dispari	R	**STATUS**
Dispari	W	*** MODE, CNTRL, SYNC1 e SYNC2**

Tabella 26-Indirizzamento dei registri dell'interfaccia seriale

11.1.8 Programmazione

Prima di iniziare una comunicazione, appena dopo aver resettato l'interfaccia, dobbiamo scrivere un byte all'indirizzo dispari, che ci permette di accedere al registro **MODE,** per determinare il tipo di comunicazione che vogliamo effettuare (sincrona o asincrona) e il formato del carattere (numero di bit d'informazione, bit di parità, tipo di parità, numero di bit di stop per comunicazione asincrona e numero di caratteri di sincronismo per comunicazione sincrona).

Se la comunicazione è sincrona, dobbiamo scrivere, sempre all'indirizzo dispari, i caratteri di sincronismo previsti in **SYNC1** ed eventualmente in **SYNC2**.

I prossimi accessi in scrittura all'indirizzo dispari ci permettono di accedere in **CNTRL** e, quindi, di abilitare il trasmettitore e/o il ricevitore, di attivare le linee di handshaking **RTS** e **DTR**, di cancellare i bit d'errore nel registro **STATUS**. Inoltre se l'interfaccia , in una comunicazione sincrona, è stata abilitata a ricevere, deve essere fissato ad 1 il bit 7 di **CNTRL** per iniziare la ricerca dei caratteri di sincronismo.

Se la comunicazione, invece, è asincrona, dopo aver caricato **MODE,** eseguendo una scrittura all'indirizzo dispari si accede direttamente a **CNTRL**. Essa ci permette di effettuare le stesse azioni descritte per la comunicazione sincrona tranne l'inserimento nel modo "hunt".

In entrambi i casi da questo momento in poi, tutti gli accessi in scrittura all'indirizzo dispari selezionano il registro **CNTRL**. In ogni istante è , comunque, possibile resettare il componente ponendo ad 1 il bit 6 di questo registro.

Esempi di sequenze di inizializzazione dell'interfaccia seriale sono mostrati nelle Tabella 27 e Tabella 28. In entrambi gli esempi si presuppone che A0 sia stato precedentemente caricato con l'indirizzo più basso associato all'interfaccia seriale.

```
move.b    #$5d,1(A0)
move.b    #$37,1(A0)
```

Tabella 27

In Tabella 27 la prima istruzione, scrivendo in **MODE**,

- imposta la trasmissione come di tipo asincrona,
- fissa il numero di bit d'informazione ad 8,
- abilita la presenza del bit di parità dispari,
- fissa ad 1 il numero di bit di stop.
- La seconda istruzione scrive nel registro **CNTRL**, essa
- cancella i bit d'errore nel registro **STATUS**,
- abilita il trasmettitore ed il ricevitore,
- attiva i segnali di handshaking **DTR** ed **RTS**.

```
move.b    #$84,1(A0)
move.b    #$16,1(A0)
move.b    #$b7,1(A0)
```

Tabella 28

La prima istuzione in Tabella 28, scrivendo in **MODE**,

- imposta la trasmissione come di tipo sincrona,
- fissa il numero di bit d'informazione a 6,
- disabilita la presenza del bit di parità,
- fissa ad 1 il numero dei caratteri di sincronismo.

La seconda istruzione memorizza il carattere di sincronismo nel registro **SYNC1**, il carattere scelto è pari a 16 esadecimale.

L'ultima scrittura avviene nel registro **CNTRL**, essa

- cancella i bit d'errore nel registro **STATUS**,
- abilita il trasmettitore ed il ricevitore,
- abilita la ricerca dei caratteri di sincronismo,
- attiva i segnali di handshaking **DTR** ed **RTS**.

Questo componente permette di realizzare operazioni di I/O sia in modo programmato che attraverso interruzioni.

In entrambi i casi, se desideriamo leggere i caratteri ricevuti, dobbiamo ogni volta controllare che i bit 3, 4 e 5 (questo solo per comunicazione asincrona) del registro **STATUS** siano pari a zero, cioè che non vi sia stato nessun errore nella trasmissione del carattere.

Nell'eseguire operazioni di I/O in modo programmato deve essere continuamente esaminato il valore di uno dei due bit meno significativi di **STATUS**.

In particolare, se vogliamo trasmettere un carattere, prima di inserirlo nel registro buffer **DATOUT**, dobbiamo verificare che il bit 0 sia pari ad 1, assicurandoci, in questo modo, che il carattere precedentemente inserito in **DATOUT** sia stato già trasferito in **TSHIFT**, evitando così un errore di *overrun*.

Invece, quando intendiamo leggere un carattere dal registro buffer **DATIN**, dobbiamo verificare che il bit 1 sia pari ad 1, questo significa che il carattere ricevuto in **RSHIFT** è stato trasferito in **DATIN**. Qualora il carattere non venga letto ed un nuovo carattere viene ricevuto, esso viene trasferito in **DATAIN** cancellando il carattere precedente e, quindi, causando un errore di *overrun*.

In alternativa le operazioni di I/O possono essere realizzate con l'ausilio delle interruzioni specificate nei parametri **COM2** e **COM3**. La routine associata alla prima interruzione deve assicurarsi di eventuali errori nella ricezione e quindi leggere il carattere da **DATIN**, mentre, quella associata all'interruzione specificata in **COM3** deve provvedere a scrivere in **DATOUT** il prossimo carattere da trasmettere.

11.1.9 Il comportamento.

In questo paragrafo verrà descritto il comportamento del dispositivo nei vari modi di funzionamento facendo implicito riferimento che le interruzioni vengono inviate al gestore delle interruzioni specificato in **COM1** e che le linee di handshaking e le linee di ricezione e trasmissione dati sono collegate al device specificato in **COM4**.

Dato che la frequenza del segnale di clock applicato all'interfaccia seriale è almeno 30 volte inferiore a quella applicata al processore, bisogna ridurre la frequenza di clock, simulata, applicata al modulo che simula l'interfaccia seriale.

Questo può essere realizzato con le seguenti operazioni:

1) attivare il menù **Device** dalla finestra principale;

2) scegliere il comando **Speed**;

3) inserire un valore maggiore di uno nella finestra che appare.

Maggiore sarà il valore inserito più bassa sarà la frequenza del segnale di clock applicata all'interfaccia. Il valore inserito indica dopo quanti colpi di clock, applicati al processore, viene mandato uno all'interfaccia.

Ovviamente, i colpi di clock cui faremo riferimento successivamente saranno quelli applicati all'interfaccia seriale e non al processore.

11.1.10 Trasmissione asincrona

Il trasmettitore è pronto a trasmettere solo dopo la sua abilitazione e dopo aver ricevuto i segnali **DSR** e **CTS**; tuttavia, fino a quando non viene inserito un carattere in **DATOUT**, sarà trasmesso il valore '1'per ogni colpo di clock.

Quando scriviamo un carattere in **DATOUT**, viene azzerato il bit 0 di **STATUS**, poi al prossimo impulso di clock:

- il valore in **DATOUT** viene copiato in **TSHIFT**,
- viene fissato ad 1 il bit 0 di **STATUS**,
- viene inviata un'interruzione sulla linea specificata in **COM3** ,
- infine viene trasmesso un bit '0', detto bit di start.

Successivamente per ogni impulso di clock viene trasmesso il bit 0 di **TSHIFT** e contemporaneamente il contenuto di questo registro viene shiftato di una posizione verso destra. Questo comportamento continua fin quando non vengono trasmessi tutti i bit d'informazione previsti in **MODE**. Se il carattere contiene più bit di quelli da noi previsti, allora i bit in eccesso non saranno presi in considerazione.

Il prossimo bit ad essere trasmesso, se esso è stato richiesto in **MODE**, è il bit di parità, con il giusto tipo di parità, altrimenti viene trasmesso un bit "1" citato come bit di stop. Se in **MODE** sono stati fisssati due bit di stop, al prossimo colpo di clock verrà trasmesso un'altro "1" riportandoci nello stato iniziale.

Se nel frattempo non è stato inserito un nuovo carattere in **DATOUT**, sarà trasmesso un "1" per ogni successivo colpo di clock.

11.1.11 Trasmissione sincrona

Dopo aver impostato la trasmissione come di tipo sincrona , dopo aver abilitato il trasmettitore e ricevuto i segnali **DSR** e **CTS**, il trasmettitore è pronto a trasmettere, ma fin quando non viene inserito un carattere in **DATOUT**, verrà posto ad 1 il bit 3 di **STATUS** e saranno trasmessi in continuazione i caratteri di sincronismo.

Precisamente il prossimo impulso di clock produce le seguenti azioni:

- copia del valore di **SYNC1** in **TSHIFT**,
- trasmissione del bit 0 di **TSHIFT**,
- shift verso destra di una posizione del valore in **TSHIFT**.

Successivamente per, ogni impulso di clock, vengono ripetute le ultime due operazioni fino alla completa trasmissione degli otto bit del primo carattere di sincronismo. Se deve essere trasmesso anche il secondo carattere di sincronismo, tutte le operazioni sono ripetute, con la differenza che in **TSHIFT** viene caricato **SYNC2** invece che **SYNC1**.

Se, nel frattempo, non è stato caricato in **DATOUT** nessun carattere, viene posto ad 1 il bit 3 di **STATUS** e vengono trasmessi di nuovo i caratteri di sicronismo previsti. In generale questo avviene ogni volta che il trasmettitore viene a trovarsi nella condizione di *underrun*.

Quando scriviamo un carattere in **DATOUT** viene azzerato il bit 0 di **STATUS** e al prossimo impulso di clock viene:

- copiato il valore di **DATOUT** in **TSHIFT**,
- posto ad 1 il bit 0 di **STATUS**,
- azzerato il bit 3 di **STATUS**,

- inviata un'interruzione sulla linea specificata in **COM3**,
- trasmesso il bit 0 di **TSHIFT**,
- shiftato verso destra di una posizione il valore in **TSHIFT**.

Le due ultime operazioni vengono ripetute ad ogni impulso di clock fin quando non vengono trasmessi tutti i bit d'informazione previsti in **MODE**. Anche in questo caso, se il carattere contiene più bit di quelli da noi previsti, i bit in eccesso non saranno presi in considerazione.

Il prossimo bit ad essere trasmesso, se esso è stato richiesto in **MODE**, è il bit di parità, con il giusto tipo di parità.

Se, nel frattempo, non è stato inserito un nuovo carattere in **DATOUT**, il trasmettitore viene a trovarsi nella condizione di *underrun* e quindi verrà posto ad 1 il bit 3 di **STATUS** e verranno trasmessi di nuovo i caratteri di sicronismo previsti.

11.1.12 Ricezione asincrona

Quando il ricevitore è stato abilitato ed è stato attivato il segnale **DSR**, esso è pronto a ricevere.

Se viene ricevuto un dato sulla linea **RX**, esso è inserito sempre nel bit più significativo di **RSHIFT**, dopo che quest'ultimo è stato shiftato di una posizione verso destra.

Il ricevitore inizia a considerare i bit in arrivo come bit d'informazione solo quando riceve il bit di start.

Una volta ricevuto l'ultimo bit d'informazione del carattere trasmesso, il valore in **RSHIFT** viene shiftato di un numero di posizioni verso destra pari a otto meno il numero di bit d'informazioni trasmesso, in modo da spostare il carattere al limite destro di **RSHIFT**.

A questo punto viene copiato il dato di **RSHIFT** in **DATIN** e se il bit 1 di **STATUS** è uno, cioè se il carattere ricevuto prima non è stato letto dal processore, viene segnalato un errore di *overrun* ponendo il bit 4 di **STATUS** ad uno.

Il prossimo bit ricevuto se presente è il bit di parità, se viene scoperto un errore di parità, esso viene segnalato ponendo ad uno il bit 3 di **STATUS**.

Successivamente il ricevitore si aspetta uno o due bit di stop, se, invece, viene ricevuto uno zero, esso segnala un errore di *framing* ponendo ad uno il bit 5 di **STATUS**.

Quando viene ricevuto l'ultimo bit di stop viene posto ad uno il bit 1 di **STATUS** ed è inviata un'interruzione del tipo specificata in **COM2**.

Quando poi il processore preleva il dato da **DATIN** il bit 1 di **STATUS** viene azzerato.

11.1.13 Ricezione sincrona

Anche in questo caso il ricevitore è pronto a ricevere, se esso è stato abilitato ed è stato attivato il segnale **DSR**.

Se viene ricevuto un dato, esso è inserito sempre nel bit più significativo di **RSHIFT** dopo che quest'ultimo è stato shiftato di una posizione verso destra.

Le differenze con la ricezione asincrona sono che non esistono nè bit di start, nè bit di stop e che, se il ricevitore è nello stato *hunt,* esso ricerca non i bit d'informazione ma i caratteri di sincronismo. Quando riconosce

i caratteri di sincronismo memorizzati all'inizio in **SYNC1** ed eventualmente in **SYNC2,** esso azzera il bit7 di **CNTRL**, il quale fa terminare la ricerca, e pone ad uno il bit 6 di **STATUS**.

Una volta ricevuto tutti i bit d'informazione e l'eventuale bit di parità e dopo aver segnalato i possibili errori di *overrun* e di *parità,* in modo identico a quanto visto per il ricevitore asincrono, viene posto ad uno il bit 1 di **STATUS** ed è inviata un'interruzione del tipo specificata in **COM2**.

Anche per la ricezione sincrona, quando il dato è prelevato dal processore da **DATIN**, viene azzerato il bit 1 di **STATUS**.

12 Dispositivo Tappo

La funzione di questo dispositivo e di rispedire in modo opportuno, i segnali che gli arrivano in ingresso, all'interfaccia seriale o al terminale seriale che li aveva inviati.

In questo modo il terminale o l'interfaccia riceve i caratteri trasmessi precedentemente.

Nel collegare ad un'interfaccia seriale un dispositivo periferico, qual'è il bloccatore, sono possibili, come abbiamo visto in precedenza, due tipi di connessioni: piena e parziale.

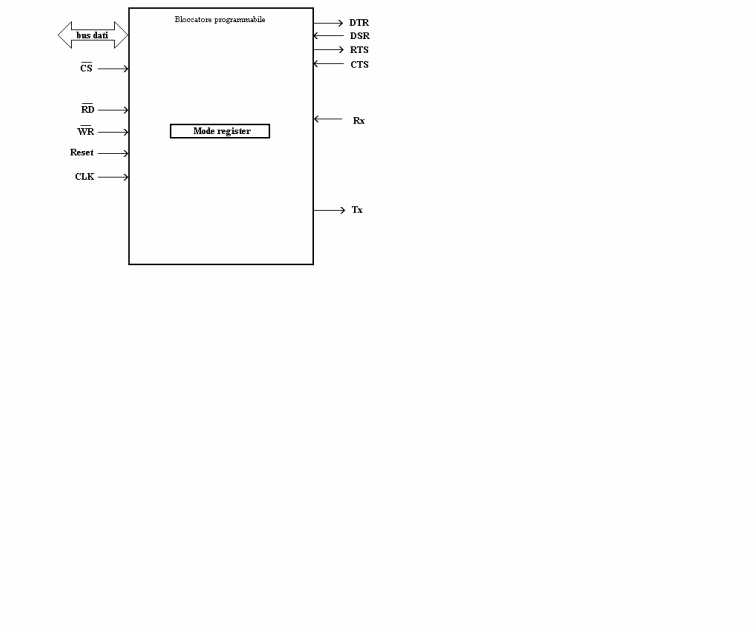


Fig. 32

E' stato simulato un bloccatore programmabile, il cui modello è presentato in Fig. 32, che permette di gestire sia connessioni parziali che piene, in funzione del valore presente nel registro di modo.

Se il componente è stato inizializzato per gestire una connessione piena, esso realizza i collegamenti tra le linee **DSR** e **DTR**, tra **CTS** ed **RTS**, e tra **Tx** e **Rx**. (figura 10). Invece, per una connessione parziale il bloccatore effettua il solo collegamento tra **Tx** ed **Rx** (figura 11).

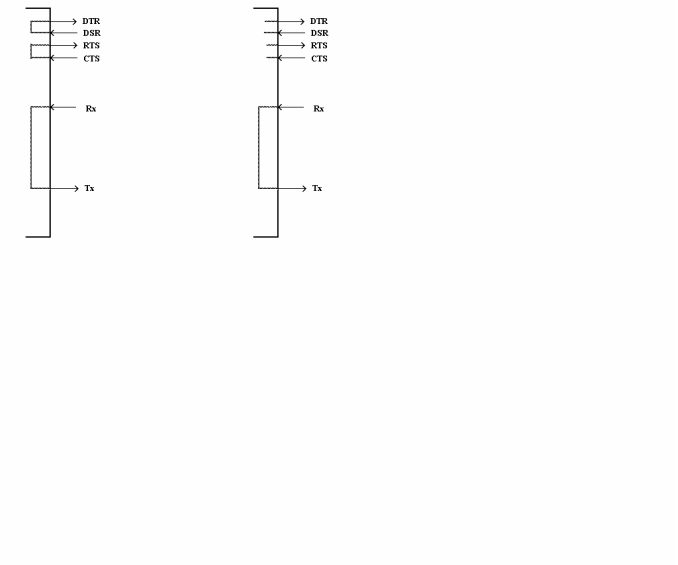


Fig. 33

Si è voluto simulare anche un bloccatore più semplice, che non è collegato al sistema bus e quindi non è programmabile, che connette le linee sempre come illustrato in Fig. 33. Il modello di questo tipo di bloccatore è mostrato in Fig. 34.

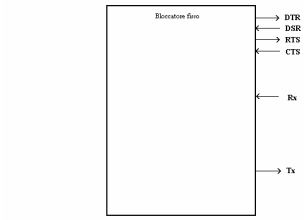


Fig. 34

Il nome del modulo che simula il bloccatore, sia esso programmabile o non, è STOPPER.

12.1.1 Collegamento al processore e ad un device

Il bloccatore programmabile presenta, rispetto all'interfaccia seriale descritta nel capitolo precedente, sostanzialmente le stesse linee verso il device e verso il processore. Quindi le proprietà per l'interfacciamento sono uguali a quelle descritte per l'interfaccia seriale.

Due esempi di collegamento sono mostrati in Fig. 35 e Fig. 36.

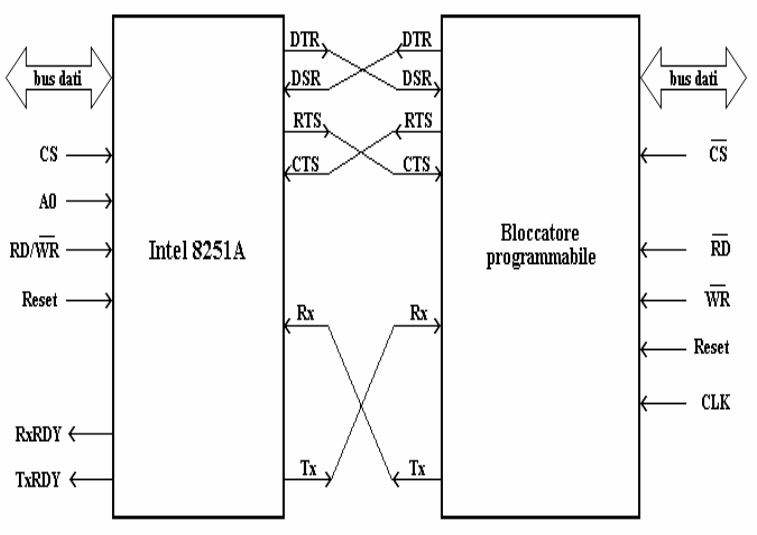


Fig. 35

I collegamenti illustrati in queste figure sono realizzabili anche con il bloccatore fisso, con la differenza che manca il suo collegamento al bus.

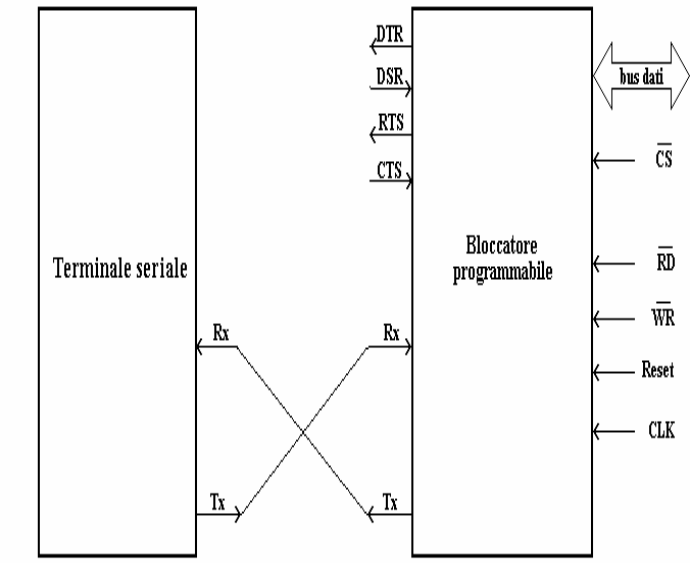


Fig. 36

12.1.2 Collegamento software

L'inserimento del bloccatore in una configurazione di ASIM si differenzia in funzione del tipo di bloccatore da simulare. In particolare, se vogliamo inserire il bloccatore fisso, dobbiamo definire solo i parametri **Nome Elemento**, **Identificatore** e **Com4**, invece, per quello programmabile bisogna fissare anche **Indirizzo1** e **BUS**.

Nome Elemento e **Identificatore** hanno sempre lo stesso significato devono essere pari rispettivamente a "*STOPPER*" e ad un *numero compreso tra 01 ed FF*.

Indirizzo1 rappresenta l'indirizzo per accedere al componente. Questo parametro permette di definire l'indirizzo sul bus indirizzi che attiva la linea **CS** e quindi che seleziona il componente.

BUS determina l'*Identificatore del bus* a cui è connesso il bloccatore programmabile. Quindi, scegliendo per **BUS** l'identificatore di un componente MMU/BUS, a cui è stato collegato il dispositivo di tipo CPU, si determina la connessione delle linee **bus dati**, **RD**,**WR** e **Reset** al processore suddetto.

COM4 specifica l'*Identificatore del Device* a cui è connesso il bloccatore, consentendo così di definire il collegamento delle linee **Tx** e **Rx** ed eventualmente delle linee **DSR**, **DTR**, **CTS** ed **RTS**, con un componente connettibile al bloccatore.

La finestra associata a STOPPER è quella in figura 15.

Fig. 37

12.1.3 Modello di programmazione

Il bloccatore programmabile ha un solo registro programmabile: il registro **MODE**. Per accedere a questo registro ad 8 bits occorre che l'indirizzo sia uguale a **Indirizzo1** e gli accessi possono essere sia in scrittura che in lettura.

In **MODE** l'unico bit che ha significato è il meno significativo. Se esso è fissato a 0 il bloccatore realizza i collegamenti tra le linee **DSR** e **DTR**, tra **CTS** ed **RTS**, e tra **Tx** e **Rx**, altrimenti, esso effettua il solo collegamento tra **Tx** ed **Rx**.

Data la semplicità della programmazione di questo dispositivo si omette l'esempio relativo.

13 Il dispositivo di collegamento tra nodi: ELABNODE.

Con i dispositivi MMU/BUS e 1to4BUSINT, si è mostrato come in ASIM sia possibile costruire macchine con più processori; questi processori possono interagire tra loro scambiandosi informazioni e dati utilizzando aree di memoria comune, cioè rispettando il cosiddetto "modello a memoria comune". Esistono tipi di architetture per macchine MIMD che fanno invece riferimento ad un "modello a scambio di messaggi" e sono realizzate utilizzando nodi di elaborazione; questi sono costituiti tipicamente da processore e memoria e sono connessi attraverso un certo numero di link; ogni comunicazione tra i vari processori avviene attraverso la rete costituta da questi link; le tipologie più comuni sono, ad esempio, quelle ad ipercubo e mesh. Il dispositivo in esame è stato introdotto per simulare con ASIM anche questo tipo di architetture.

ELABNODE è un dispositivo che presenta quattro link in ingresso(I1, I2, I3, I4) e quattro in uscita (O1, O2, O3, O4); ad ogni link è associato un registro a trentadue bit che è indicato con lo stesso nome del link. Un'operazione di trasferimento dati consiste nell'inviare o ricevere il contenuto di uno di questi registri. La sincronizzazione nel trasferimento di un dato avviene sfruttando un registro di controllo/stato (CN) ad otto bit. I quattro bit meno significativi sono associati ai link in ingresso e, se alti, assumono il significato di dato pronto; i quattro bit più significativi sono associati ai link in uscita e, se alti, assumono il significato di link occupato.

ELABNODE può essere connesso ad un bus, occupando gli indirizzi da FFFFFFEF a FFFFFFFF; al primo indirizzo si può accedere solo con un'operazione di scrittura o lettura sul byte e corrisponde al registro CN; Inoltre è possibile accedere, solo con un'operazione di scrittura o lettura su doppia parola, agli indirizzi: FFFFFFF0, FFFFFFF4, FFFFFFF8, FFFFFFFC; a questi corrispondono, in lettura, i registri I1, I2, I3, I4 ed, in scrittura, i registri O1, O2, O3, O4.

ELABNODE può essere connesso anche ad un dispositivo per la gestione delle interruzioni; ad ogni link di ingresso corrisponde una differente linea di interruzione. Quando, ad esempio, sul link I1 arriva un dato, il bit 0 (corrispondente ad I1) del registro CN va ad 1 e viene inviata l'interruzione prevista; quando il processore legge il dato da I1, ELABNODE segnala al mittente l' avvenuta ricezione e quindi la disponibilità a ricevere un nuovo dato. Inoltre, riporta a 0 il bit 0 di CN.

Per inviare un dato, ad esempio sul link O1, il processore deve innanzitutto controllare che il link non sia occupato da un precedente trasferimento, cioè deve verificare che il bit 4 (corrispondente ad O1) di CN non sia 1; poi può scrivere il dato che ELABNODE provvede ad inviare dopo aver posto ad 1 il bit 4 di CN; tale bit torna a 0 a trasferimento completato segnalando così la disponibilità ad inviare un nuovo dato.

Vi sono due possibili modalità nell'invio dei dati: la prima tiene conto della tempificazione e prevede l'invio del dato durante il primo colpo di clock successivo alla scrittura (questo tipo di collegamento sarà detto, nel seguito, "tempificato"); la seconda non tiene conto della tempificazione e prevede l' invio del dato immediatamente dopo l'operazione di scrittura. Nel primo caso è possibile, variando la velocità con il comando *Velocità* del menù *Schedulatore*, simulare link con diverso tempo di trasferimento.

La finestra associata ad ELABNODE (Fig. 38) mostra il contenuto di tutti i registri; questi possono essere direttamente modificati con il comando *Modifica Valore* del menù *Nodo*. All'accensione della macchina da simulare, tutti i registri sono al valore zero.

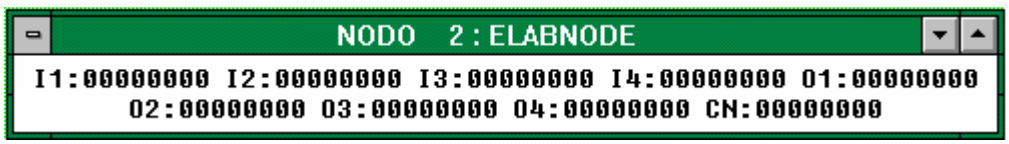


Fig. 38

Per costruire un nodo di elaborazione occorre definire almeno tre componenti: una CPU, un MMU/BUS ed un ELABNODE; si è detto almeno tre, perché nulla vieta di connettere al bus un numero qualsiasi di dispositivi anche se, tipicamente, il nodo di elaborazione è costituito da processore, memoria e link di connessione. Poiché gli indirizzi associati ai registri di ELABNODE sono fissi, non è possibile connettere ad un bus più di un dispositivo del tipo in esame. Si è già visto come definire CPU e MMU/BUS; per quanto riguarda ELABNODE occorre specificare i parametri che compaiono in una finestra di dialogo del tipo di quella mostrata in figura 3 cui si accede selezionando il comando *Aggiungi Nodo* del menù *Configura*.

Il "Nome Elemento" è ovviamente ELABNODE. "Indirizzo 1" e "Indirizzo 2" definiscono le interruzioni connesse ai link in questo modo:

quattro cifre meno significative di "Indirizzo 1" -> interruzione associata ad I1;

quattro cifre più significative di "Indirizzo 1" -> interruzione associata ad I2;

quattro cifre meno significative di "Indirizzo 2" -> interruzione associata ad I3;

quattro cifre più significative di "Indirizzo 2" -> interruzione associata ad I4.

Il significato delle quattro cifre è quello più volte presentato.

"BUS" consente di definire il bus (due cifre meno significative) e il gestore delle interruzioni (due cifre più significative) cui ELABNODE è collegato.

"Com1", "Com2", "Com3", "Com4" consentono di definire i dispositivi cui sono collegati i quattro link di uscita. Ad esempio, le due cifre meno significative di "Com1" sono destinate all'Identificatore del dispositivo da collegare ad O1; la terza cifra di "Com1" indica a quale link di ingresso del dispositivo a valle è connesso O1; i valori ammessi sono:

0 O1 è connesso al link I1 del dispositivo a valle;

1 O1 è connesso al link I2 del dispositivo a valle;

2 O1 è connesso al link I3 del dispositivo a valle;

3 O1 è connesso al link I4 del dispositivo a valle.

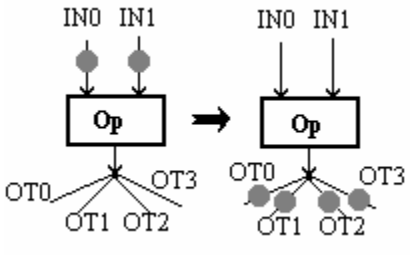
Infine la cifra più significativa di "Com1" consente di specificare se il collegamento è tempificato (cifra = 1) o non tempificato (cifra = 0).

Per "Com2", "Com3" e "Com4" vale quanto detto per "Com1" con riferimento, anziché ad O1, ad O2, O3 ed O4 rispettivamente.

Nell'attuale versione di ASIM i dispositivi collegabili ad un link sono ELABNODE e ARTMNODE.

14 Il dispositivo nodo "data flow": ARTMNODE.

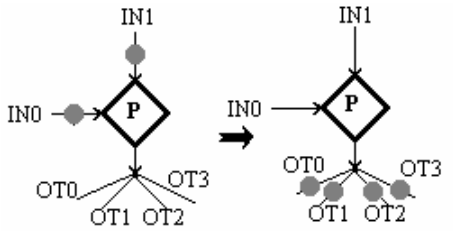
ASIM è un ambiente che consente di simulare anche architetture non Von Neuman come le macchine Data Flow. Il dispositivo ARTMNODE è stato introdotto per mostrare questa possibilità; esso definisce un nodo in grado di eseguire un operazione aritmetica su interi espressi su trentadue bit.



Codice	Operatore	IN1 cost
00	+	no
01	+	si
02	*	no
03	*	si
04	/	no
05	/	si

Fig. 39

Per ogni nodo è possibile definire l'operazione che esso deve svolgere quando tutti gli operandi previsti si rendono disponibili; più nodi possono essere connessi tra loro per eseguire espressioni algebriche; grazie alla presenza di funzioni per il confronto, porte TRUE e FALSE e funzioni di Switch e Merge, è possibile definire macchine data flow in grado di eseguire semplici algoritmi; infine, essendo possibile collegare tra loro ELABNODE e ARTMNODE, si possono costruire macchine miste, combinazione di architetture convenzionali e data flow.



Il token in uscita è booleano: true se IN0 P IN1 è vero, false altrimenti. Se IN1 è costante, il token corrispondente è considerato sempre presente e basta l' arrivo di IN0 per lo scatto.

Codice	Predicato	IN1 cost
10	=	no
11	=	si
12	<	no
13	<	si
14	<=	no
15	<=	si
16	>	no
17	>	si
18	>=	no
19	>=	si

Fig. 40

ARTMNODE presenta quattro registri collegati ai link in ingresso (IN0, IN1, IN2, IN3) e quattro registri collegati ai link in uscita (OT0, OT1, OT2, OT3); il numero di registri in ingresso, e quindi di link, utili dipende dalla funzione

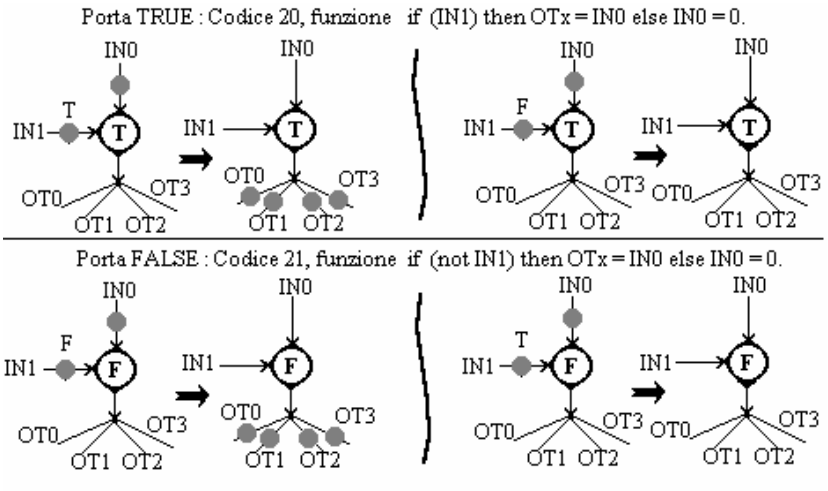


Fig. 41

svolta dal nodo. Ad esempio, se la funzione f prevede due operandi, solo IN0 e IN1 vengono utilizzati; quando sono arrivati entrambi i dati (token) sui link corrispondenti, viene eseguita l' operazione f(IN0,IN1); il risultato viene contemporaneamente scritto sui registri di uscita ed inviato ad eventuali nodi connessi ai link; infine IN0 e IN1 vengono riportati al valore 0. Nel caso in cui una funzione prevede che una variabile sia costante, il registro che la contiene non viene azzerato. Se un nuovo dato arriva in un registro di ingresso prima che il dato precedente sia stato elaborato, il dato precedente viene perso. Le regole di scatto e le funzioni disponibili (con relativo codice) sono riportate nelle figure 13, 14, 15, 16.

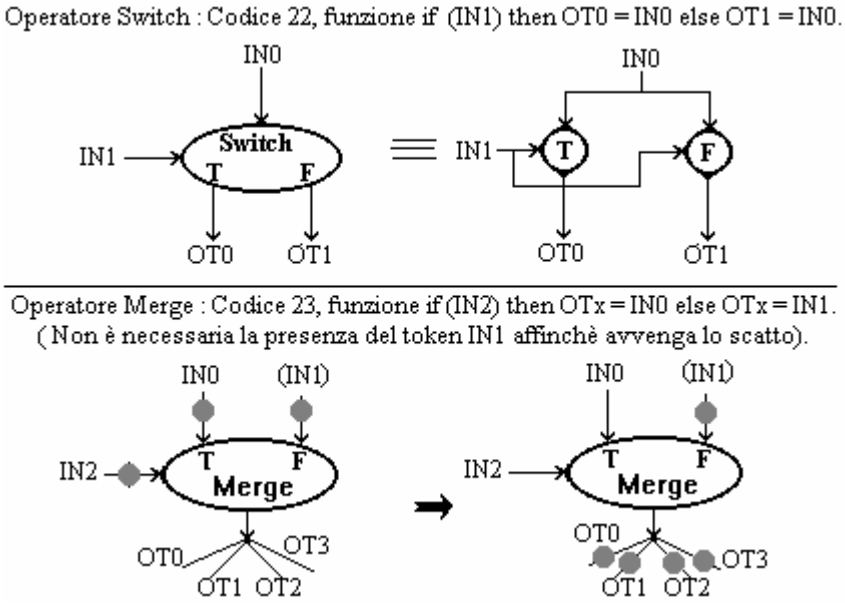


Fig. 42

La finestra associata ad ARTMNODE (17Fig. 43) riporta la funzione svolta dal nodo e tutti i registri (compresi quelli non utilizzati); il contenuto dei registri può essere modificato utilizzando il comando *Modifica Valore* del menù *Nodo*; si osservi che la modifica diretta è equivalente all' arrivo del dato sul link corrispondente al registro modificato, cioè produce gli stessi effetti.

Come nel caso di ELABNODE, vi sono due possibili modalità nell' invio dei dati (token): la prima tiene conto della tempificazione e prevede l' invio del dato durante il primo colpo di clock successivo alla scrittura sul registro di uscita; la seconda non tiene conto della tempificazione e prevede l' invio del dato immediatamente dopo l'

operazione di scrittura. Nel primo caso è possibile, variando la velocità con il comando *Velocità* del menù *Schedulatore*, simulare link con diverso tempo di trasferimento.

Quando si simula un algoritmo utilizzando una macchina data flow e si vuole vedere l'esecuzione dell'algoritmo passo passo, è necessario tempificate qualche link di uno o più nodi. Solo in tal caso è, infatti, possibile utilizzare il comando *Passo* del menù *Schedulatore* più volte; altrimenti, l'intero algoritmo verrà eseguito in un sol passo. Se non viene tempificato nessun link si DEVE prevedere per l'algoritmo una condizione di terminazione, altrimenti non lo si potrà più fermare una volta avviato.

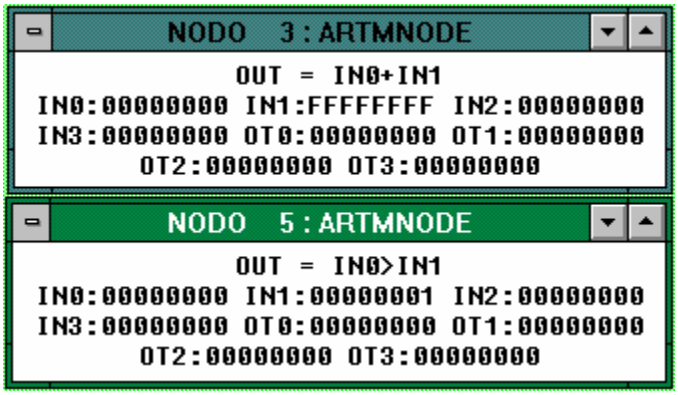


Fig. 43

Per definire un ARTMNODE occorre specificare i parametri che compaiono nella finestra di configurazione di ASIM e a cui si accede selezionando il comando *Aggiungi Nodo* del menù *Configura*.

Il "Nome Elemento" è ARTMNODE. "Indirizzo 1" consente di specificare la funzione che deve svolgere il nodo; il codice che corrisponde a ciascuna funzione è presentato nelle figure 13, 14, 15, 16. "BUS" non è utilizzato da questo dispositivo. "Com1", "Com2", "Com3", "Com4" consentono di definire i dispositivi cui sono collegati i quattro link di uscita. Ad esempio, le due cifre meno significative di "Com1" sono destinate all'Identificatore del dispositivo da collegare ad OT0; la terza cifra di "Com1" indica a quale link di ingresso del dispositivo a valle è connesso OT0; i valori ammessi sono:

0 OT0 è connesso al link IN0 del dispositivo a valle;

1 OT0 è connesso al link IN1 del dispositivo a valle;

2 OT0 è connesso al link IN2 del dispositivo a valle;

3 OT0 è connesso al link IN3 del dispositivo a valle.

Infine la cifra più significativa di "Com1" consente di specificare se il collegamento è tempificato (cifra = 1) o non tempificato (cifra = 0).

Per "Com2", "Com3" e "Com4" vale quanto detto per "Com1" con riferimento, anzichè ad OT0, ad OT1, OT2 ed OT3 rispettivamente.

Parte II

Esempi di configurazioni ASIM

Parte III

Progetti in ASIM

Parte IV Assembler MC68000