## Il dispositivo DMA

Un processore esegue istruzioni ad una velocità legata al suo clock (ad esempio n cicli v.Neumann/k cicli di clock con k>n). Il numero di istruzioni al secondo è, allo stato attuale della tecnologia, dell'ordine di Giga istruzioni/sec ($10^9$ GIPS) per processori commerciali. La velocità del processore è quasi sempre, a parte taluni casi particolari, molto maggiore rispetto a quella di un dispositivo, specialmente di dispositivi elettromeccanici quali le stampanti, i plotter, ecc.

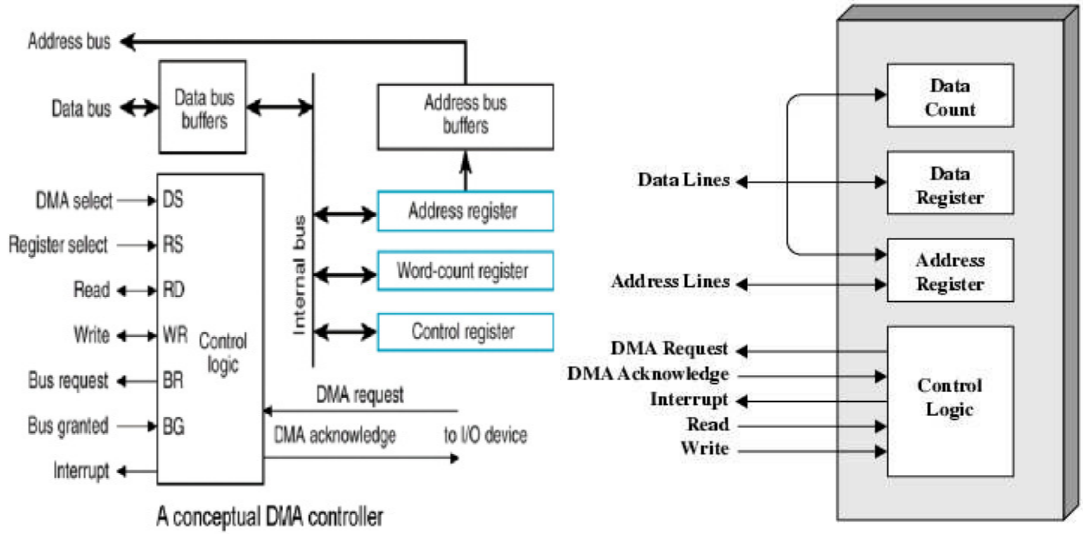
Le operazioni di trasferimento da e per un device/periferica, sono caratterizzate da un tempo di setup iniziale, necessario ad avviare l'operazione stessa e indipendente dal numero di bit o byte da trasferire; da un tempo di trasferimento del dato/dati; da un tempo di chiusura dell'operazione. I trasferimenti possono essere di tipo bit oriented, e cioè il trasferimento consiste nello scambio di una stringa di bit di lunghezza n spesso variabile ad ogni trasmissione, o di tipo byte oriented (o a carattere) in cui il trasferimento consiste nello scambio di n byte (singolo trasferimento di byte o trasferimento di n byte). Un trasferimento può essere effettuato o in modalità "polling" o in mediante "interrupt". Nel primo caso il processore è impegnato per tutta la durata del trasferimento o in attesa che esso avvenga (busy wait), nel secondo caso eventi asincroni interrompono il processore solo quando l'operazione è pronta ad essere eseguita. Da quanto detto è chiaro che la soluzione di busy wait è, particolarmente, onerosa se si gestisce una periferica lenta che tra un trasferimento e l'altro, o per un trasferimento, impiega molto tempo. La soluzione che fa uso degli interrupt può essere critica se il trasferimento interessa una periferica veloce in quanto la latenza del servizio delle interruzioni può essere di valore comparabile o maggiore dl tempo di reazione della periferica.

Supponendo che un processore operi ad una velocità di 100 M istruzioni/sec e supponendo che esso esegua un'istruzione/ciclo di clock, il suo clock avrà una frequenza di 100 MHz e un tempo di ciclo di 0,01 μsec.

Operazioni a carattere (ad es. processore-modem asincrono oppure processore-stampante a caratteri):

Ipotesi: periferica che opera con una velocità di trasferimento di 10000 car/sec (ovvero di 1 carattere/10msec) e processore a 100 MHz. In 1 sec il processore esegue circa 100 M istruzioni per cui, se si opera in polling, 10 msec di attesa implicano la mancata esecuzione di 10000 istruzioni.

Il dispositivo Direct Memory Access (DMA) è un dispositivo in grado di operare da master di bus, in grado di gestire cicli bus di trasferimento dati ( in sostituzione del processore). Esso viene utilizzato per aumentare le prestazioni di un sistema di calcolo gestendo, in sostituzione del processore ma in modo più efficiente, le operazioni di trasferimento, consentendo, nel contempo, al processore di operare in parallelo. La maggiore efficienza è dovuta al fatto che mentre un processore per eseguire un trasferimento deve eseguire un'istruzione (ad es. un codice di move) attraverso una fase di fetch ed execute, un DMA gestisce il trasferimento utilizzando Hw specializzato appositamente progettato per realizzare tale unica funzione.




In un'operazione di DMA il processore attiva la linea R/W per indicare la lettura o scrittura, pone sul bus dati l'indirizzo fisico del dispositivo di I/O interessato e, successivamente, l'indirizzo iniziale in memoria del blocco dati coinvolto nell'operazione (da dove leggere o dove scrivere i dati) e la quantità di dati da trasferire, avviando in tal modo l'operazione di trasferimento del blocco dati. Il processore può, a questo punto, eseguire altri codici operativi (sempre che non si creino conflitti di accesso sul bus) mentre il controllore DMA si occupa del trasferimento dei dati colloquiando direttamente con la memoria centrale. Alla fine del trasferimento del blocco il controllore DMA invia un interrupt alla CPU al fine di avviare le operazioni di chiusura dell'operazione. Durante il trasferimento dati il DMA può accedere al canale dati o una parola alla volta, sottraendo di tanto in tanto alla CPU il controllo sul canale (cycle stealing) oppure per blocchi, prendendo possesso del canale per una serie di trasferimenti (burst mode).

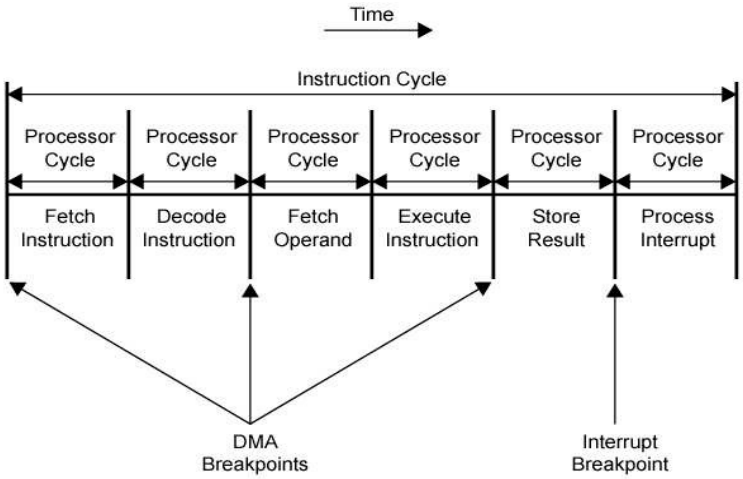
La CPU è bloccata in entrambi i casi, ma il burst mode è più efficace perché la latenza di acquisizione del canale è comunque un'operazione onerosa.

Nel Caso di un trasferimento dati in cycle stealing il controllore DMA prende possesso del bus per un ciclo necessario a trasferire una parola (word) di dati. In tale tipo di operazione la CPU non cambia contesto (non è una interruzione), il suo ciclo viene temporaneamente ritardato prima che acceda al bus Ad esempio, prima del caricamento di un dato e/o operando o di una scrittura. Tale sospensione crea, di fatto, un rallentamento del processore ma non così tanto come nel caso in cui sia la CPU stessa ad occuparsi del trasferimento dati.

Breakpoint di DMA e di Interrupt durante un ciclo di istruzione

Un ciclo istruzione un processore è realizzato mediante la successione di fasi elaborative eseguite ad ogni ciclo di clock (più cicli di clock (o, in taluni casi più fasi di clock) realizzano un ciclo istruzione. Nella figura seguente si evidenzia il caso di un processore che esegue un ciclo v Neumann in 6 fasi (fetch istruzione, decodifica, fetch operando, esecuzione istruzione, memorizzazione risultato, controlla presenza interuzioni). In essa sono indicati i possibili punti di interruzione del ciclo (breakpoint) per operazioni di DMA e di interruzioni. Come è noto, la sospensione del ciclo richiede il salvataggio del contesto elaborativo e, per mantenere il contenuto informativo associato allo stato limitato e trattabile (anche ai fini delle prestazioni,

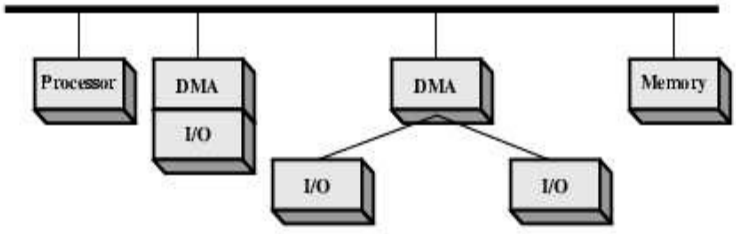
dovendo il contesto essere salvato), si individuano punti ben precisi del ciclo (tipicamente posti alla fine di una fase operativa) come quelli indicati nell'esempio.




Un sistema può essere architettato in tanti modi differenti a seconda delle risorse hardware disponibili e impiegate. Elemento critico per una architettura a processore è il bus. Si possono definire architetture semplici basate su singolo bus o architetture più complesse basate su più bus. I bus possono essere specializzati per operazioni di I/O verso device lenti/veloci o per trasferimenti verso risorse quali la memoria centrale.

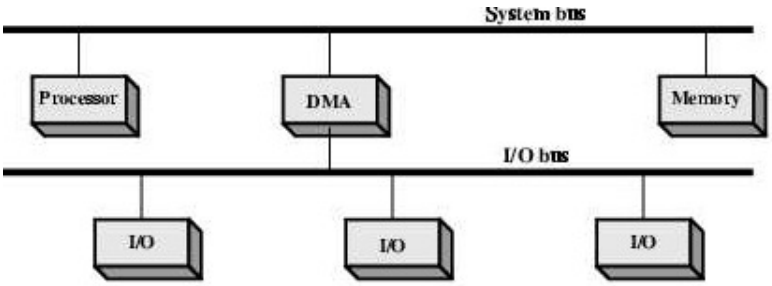
Ad esempio nell'architettura a semplice bus singolo di seguito riportata, la connessione fra DMA e I/O è fatta per tamite il bus: in tal caso il controller DMA legge o scrive dal device di I/O e scrive o legge verso la memoria. Ogni trasferimento usa il bus due volte: da I/O a DMA e poi da DMA alla memoria e la CPU perde il possesso del bus due volte.

Una soluzione alternativa più efficiente, basata sempre sull'uso di un Bus singolo, fa ricorso a un controllore DMA integrato con I/O e in grado di controllare più di un dispositivo. In tal caso ogni trasferimento usa il bus una volta (da DMA a memoria) e la CPU perde il controllo del bus una sola volta.



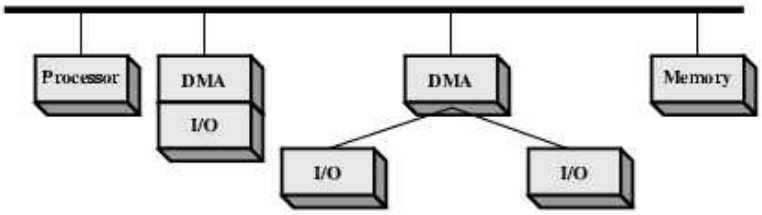

(b) Single-bus, Integrated DMA-I/O

Architetture più efficienti fanno uso di un Bus di I/O separato come nella figura di seguito mostrata. In tale tipo di architettura il DMA necessita di una sola interfaccia di I/O e per essa ogni trasferimento usa il bus di sistema una sola volta (da DMA a memoria), la CPU perde il controllo del bus una sola volta.




(c) I/O bus

Come nel caso del bus singolo, una soluzione più efficiente è data dalla seguente soluzione.




(b) Single-bus, Integrated DMA-I/O

## Caratteristiche del bus 68000

Il processore MC68000 utilizza un bus a 32 bit di tipo asincrono. Il controllo del bus è realizzato mediate due gruppi di segnali tutti 0-attivi: segnali di controllo del bus asincrono (4 segnali in out e 1 segnale in in) e segnali di arbitraggio del bus (1 segnale in in e 2 segnali in out). Mediante il primo gruppo di segnali il processore sviluppa cicli di bus (read, write, read-modify-write) mediante i quali effettuare il trasferimento dei dati da e per i device ad esso connessi. Mediante il secondo gruppo di segnali viene gestita la condivisione del bus rispetto ad altri dispositivi aventi capacità di divenire master di bus. La logica dell'arbitraggio è tale da favorire prima possibile la potenziale richiesta di bus. Il significato di tali segnali è di seguito spiegato:

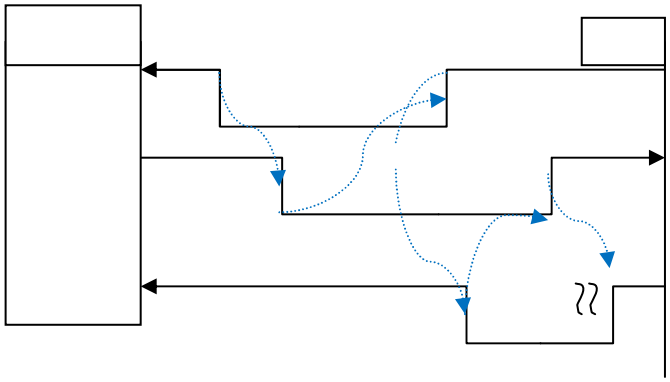
Asynchronous Bus Control

- !AS (!address strobe-out): segnale a livelli che indica che nel bus address è presente un indirizzo valido;
- R/!W (read/!write-out): segnale a livelli che indica la direzione del trasferimento;
- !UDS (!upper data strobe-out): segnale a livelli che indica la validità dei dati presenti in D8-D15;
- !LDS (!lower data strobe-out): segnale a livelli che indica la validità dei dati presenti in D0-D7;
- !DTACK (!data acknowledge-in): segnale a livelli che indica l'avvenuto trasferimento dei dati.
-

Bus Arbitration Control

- !BR (!bus request-in): segnale che indica al processore la richiesta di bus da parte di un dispositivo. Tale segnale va posto in or di connessione se a richiedere il bus sono più dispositivi;
- !BG (!bus grant-out): segnale che indica a tutti i device richiedenti il bus che il processore rilascerà il bus alla fine del ciclo corrente;
- !BGACK (!bus grant acknowledge-in): segnale che indica che un device è divenuto il bus master e lo sta gestendo.

In genere, si possono presentare due tipologie di architetture per sistemi che prevedono la gestione in competizione del bus: arbitraggio controllato da uno dei processori configurato come master; ricorso ad un sistema di arbitraggio esterno che, secondo politiche statiche o dinamiche, assegna il bus di sistema ai vari processori e device con capacità di essere master di sistema.



1. Il dispositivo richiede il possesso del bus al processore.
2. Il processore da l'assenso alla gestione del bus
3. Il dispositivo elimina la richiesta (0->1) e asserta BGACK dichiarando la presa di possesso del bus (1->0)
4. Il processore ripristina BG (0->1)
5. Il dispositivo ripristina BGACK (0-1) segnalando così il rilascio del bus

| | |
|---|---|
| Name | I8237DMA |
| Type | Device |
| Identif | Intero ($01..$FF) che identifica univocamente l'oggetto in una configurazione |
| Address1 | Indirizzo base del dispositivo |
| Address2 | Indirizzo base+$F (DMA occupa 16 locazioni di memoria) |
| BUS | Ident. del bus a cui il dispositivo è connesso |
| COM1 | Ident. della CPU master per il trasferimento |
| COM2 | \|vettore (b15-b8)\|priorità (b7-b4)\|linea Int (b3-b0)\| |
| COM3 | Ident. Gestore interruzioni |
| COM4 | bit0-3 e bit4-7 ident. dispositivi connessi al canale 0 e 1 |

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 in grado di operare parallelamente al processore e gestire, mediante hardware specializzato le operazioni di trasferimento. Un dispositivo DMA controller è, infatti, capace, opportunamente programmato, di diventare il master del bus e supervisionare un trasferimento tra memoria e memoria o 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.

Come dispositivo commerciale da cui derivare il DMA didattico inserito nell'ambiente ASIM, si è scelto l'Intel 8237, un semplice controllore programmabile per l'accesso diretto in memoria a 4 canali impiegato nella famiglia Intel 808x e oggi disponibile come "Core" (C8237 core by Altera), specificato in VHDL o VERILOG e utilizzabile per lo sviluppo di ASIC e FPGA custom, unitamente a altri componenti MSI e VLSI.

Il DMA controller è organizzato in moduli funzionali così come rappresentato nel seguente diagramma a blocchi.

Il modulo Timing & Control genera la tempificazione interna al chip e dei segnali di controllo verso l'esterno. La tempificazione è derivata dal clock collegato esternamente al chip secondo i seguenti due tipi di ciclo: ciclo di idle (Si) e cicli attivi (S0, S1, S2, S3, S4). Tali cicli caratterizzano i trasferimenti verso dispositivi di I/O. I trasferimenti memoria-memoria occupano due canali alla volta e richiedono prima un trasferimento dalla memoria sorgente verso un registro temporaneo e un successivo trasferimento dal registro temporaneo alla memoria destinazione, utilizzando 8 (4+4) cicli di trasferimento (S11, S12, S13, S14) per la prima metà del trasferimento e (S21, S22, S23, S24) per la seconda metà. Ciascuno stato Si dura un periodo di clock.

Modulo di Fixed Priority & Rotating Priority Logic

In caso di competizione su due o più canali, il controllore DMA può selezionare il canale da attivare in base ad uno schema prioritario fisso, in cui l'ordine prioritario è legato alla posizione del canale (3 più prioritario e 0 meno prioritario) o dinamico a priorità ruotante (round-robin) in cui l'ultimo canale che ha ricevuto il servizio passa in posizione meno prioritaria.

I registri del DMA

Il DMA controller contiene al suo interno una memoria di 344 bit organizzata sotto forma di registri. L'accesso ai registri può avvenire in lettura/scrittura o, per taluni di essi o in sola lettura o scrittura. La selezione dei registri è fatta mediante le quattro linee A3-A0, i segnali di lettura (IORN) e scrittura (IOWN) attivi negati, solo se il chip è abilitato assertando il chip select negato (CSN=0).

Command Register: registro di sola scrittura posto all'indirizzo $8

Il registro CNTR controlla le operazioni del DMA, esso è programmato dal microprocessore ed è resettato dal segnale di Reset o dall'istruzione Master Clear.

Le richieste di trasferimento in arrivo sui vari canali sono gestite secondo logica prioritaria fissa o di tipo round-robin. Il trasferimento dei dati può avvenire in uno dei seguenti 4 modi:

- **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;
- **Block**: posta e soddisfatta la richiesta del bus, sarà effettuato l'intero trasferimento del blocco;
- **Demand**: simile al modo **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;
- **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 max 64 Kbyte. Alla fine di ogni trasferimento, la linea di fine conteggio segnala al processore (o alla periferica che aveva attivato) la fine del trasferimento).

Il DMA consente, inoltre, l'auto inizializzazione del conteggio associato al trasferimento, le operazioni di trasferimento di tipo memoria-memoria; la richiesta di DMA programmata e l'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 limitati a **single** e **block**. In Fig. **Errore. Nel documento non esiste testo dello stile specificato.**-1 Schema di collegamento del DMA, è mostrato un esempio di interconnessione del DMA con un sistema basato sul processore M68000.

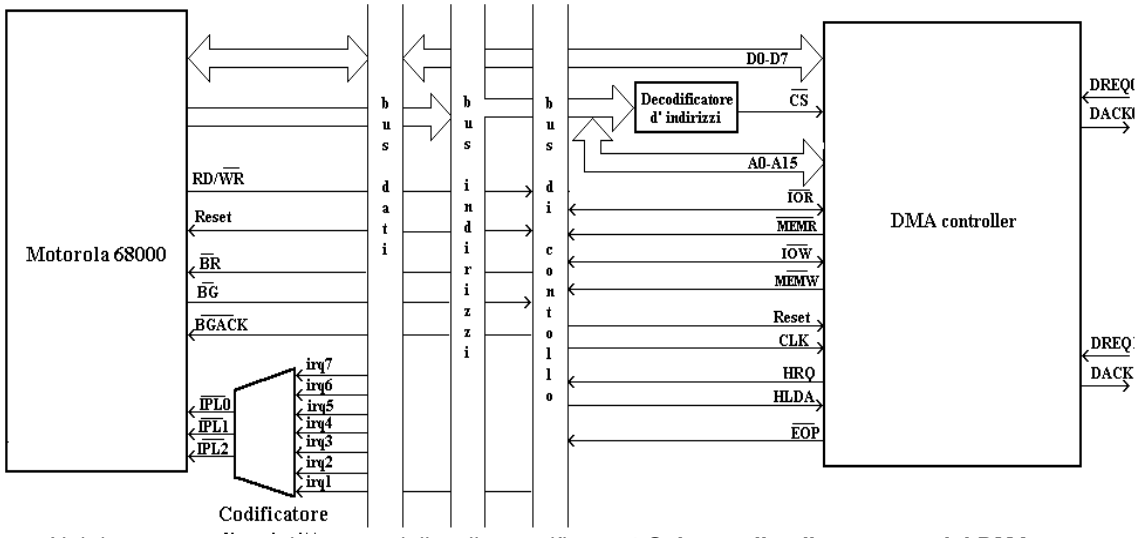



**Fig.** Errore. Nel documento non esiste testo dello stile specificato.**-1 Schema di collegamento del DMA**

Un DMA controller in una configurazione ASIM si collega d un lato al processore, che avendo accesso ai suoi registri, attraverso il bus di sistema ne permette la programmazione delle operazioni e le periferiche con cui effettuare operazioni di trasferimento da e per la memoria.

## Collegamento al processore

Le linee di collegamento al processore, sono mostrate sulla sinistra dello schema di Fig. **Errore. Nel documento non esiste testo dello stile specificato.**-1 Schema di collegamento del DMA. In particolare, **D0-D7** sono 8 linee che vanno connesse al bus dati per il trasferimento dei dati da e verso il componente; **CS** è utilizzata per la selezione del dispositivo mentre la selezione, dei registri interni è effettuata mediante le linee **A0-A3**del bus indirizzi e dai segnali di lettura-scrittura sul componente: **IOW** e **IOR**. Il controllore, per eseguire un trasferimento, deve attivare 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 di reset che esegue il posizionamento al valore 0 di tutti i registri del controllore. La linea **HRQ** (Hold Request) è adoperata per spedire una richiesta di controllo del sistema bus. Essa normalmente è applicata all'ingresso HOLD della CPU. Un segnale, in arrivo dalla CPU, sulla linea **HLDA** (Hold Ack) indica che è stato acquisito il sistema bus. La linea d'interruzione **EOP** trasmette al processore, o a un eventuale gestore delle interruzioni, un interruzione per avvisare che il trasferimento di un blocco di memoria è stato completato.

## Collegamento ad un device

Sulla destra dello schema in Fig. **Errore. Nel documento non esiste testo dello stile specificato.**-1 Schema di collegamento del DMA, sono mostrate le linee fisiche che collegano il DMA controller alle periferiche. In particolare, **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**. Le linee **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".

## Configurazione ASIM del DMA

In Fig. **Errore. Nel documento non esiste testo dello stile specificato.**-2 Configurazione del DMA, è mostrato lo schema concettuale di interconnessione del dispositivo DMA industriale in un sistema. Per inserire il componente in una configurazione ASIM, 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 FFe viene utilizzato dal programma per riferire il dispositivo;

**Indirizzo1** rappresenta l'indirizzo base del dispositivo per la selezione dei suoi registri;

**Indirizzo2**è dato da **Indirizzo1** + $F, essendo lo spazio indirizzabile di 16 byte.

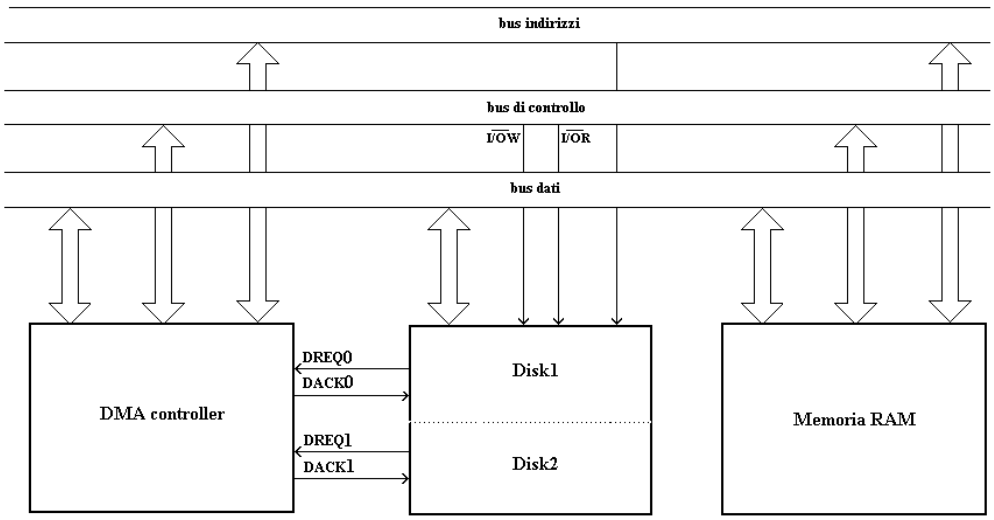



**Fig.** Errore. Nel documento non esiste testo dello stile specificato.**-2 Configurazione del DMA**

**BUS:** determina l'Identificatore del bus a cui è connesso il dispositivo (per cui, si simula il collegamento a un componente MMU/BUS delle linee D0-D7, A0-A7, IOR, IOW, MEMR, MEMW, CLK e Reset del DMA).

**COM1:** definisce l'Identificatore della CPU alla quale richiedere il controllo del bus mediante un handshaking (si simula il collegamento logico delle linee HRQ e HLDA).

**COM2:** specifica del tipo di interruzione trasmessa al gestore delle interruzioni (CPU o PIC), indicato in COM3, alla fine del conteggio di un trasferimento di 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. I due parametri COM2 e COM3 determinano, quindi, la connessione logica della linea d'interruzioni EOP al processore o al PIC.

**COM4:** le 2 cifre meno significative e le 2 più significative indicano, rispettivamente, l'identificatore della periferica collegata al canale 0 e al canale 1. Specificando il valore in COM4 si definisce, di fatto, a quali componenti collegare logicamente le coppie di linee (DREQ0, DACK0) e (DREQ1, DACK1).

## Modello di programmazione

La finestra di programmazione associata al dispositivo I8237DMA ed registri programmabili del controllore sono riportati inFig. **Errore. Nel documento non esiste testo dello stile specificato.**-3 Modello di programmazione del DMA.

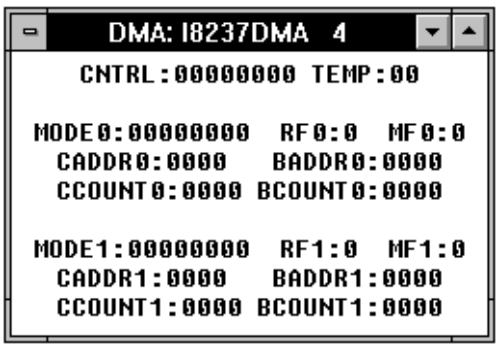



**Fig.** Errore. Nel documento non esiste testo dello stile specificato.**-3 Modello di programmazione del DMA**

Il dispositivo è costituito da due canali DMA simmetrici programmabili mediante i registri BADDR (indirizzo base del trasferimento), CADDR (indirizzo corrente del trasferimento), BCOUNT (valore base del

conteggio espresso come numero di byte da trasferire), CCOUNT (valore corrente del conteggio). A tali registri si aggiunge il registro MODE che codifica la modalità del trasferimento il flag RF che consente di avviare via software il trasferimento (senza la richiesta di una periferica) e MF per mascherare il trasferimento sul canale. Il registro di controllo/stato CNTRL è comune ai due canali, così come il registro TEMP utilizzato per appoggiare il dato da trasferire nei trasferimenti memoria-memoria. Di seguito si riporta il dettaglio di programmazione di detti registri:

- **CADDR0** e **CADDR1** (indirizzo corrente del trasferimento) del DMA: hanno parallelismo 16 bit e contengono l'indirizzo della locazione di memoria interessata 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** e 2 per **CADDR1**.

- **BADDR0** e **BADDR1** sono i registri indirizzo di base ed hanno la funzione di memorizzare 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.

- **CCOUNT0** e **CCOUNT1** sono i registri di conteggio correnti, anch'essi a 16 bit, memorizzano il numero di byte da trasferire; 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 registri a 16 bit che memorizzano il 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.

- **MODE0** e **MODE1** sono i registri che contengono le informazioni relative al modo di funzionamento dei rispettivi canali. Essi sono di sola 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** è illustrato nella Tabella Errore. *Nel documento non esiste* **testo dello stile specificato.**-1-Significato dei bit nei registri MODE**.**

- **RF0** e **RF1** sono i flag dove indirizzare le 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 | Se posto a 0 selezione il canale 0 se posto a 1 il canale 1 |
| 1-2 | Bit 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'auto inizializzazione, cioè al termine del conteggio i registri indirizzo e conteggio corrente 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 | Bit non utilizzato |
| 7 | determina il modo del trasferimento: 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** Errore. Nel documento non esiste testo dello stile specificato.**-1-Significato dei bit nei registri MODE**

- **MF0** e **MF1** del dispositivo, accessibili in sola scrittura, mascherano le richieste dei rispettivi canali se il mask flag di quel canale è posto ad 1. L'indirizzo relativo di entrambi i flag è $A, il canale selezionato, come prima in base al valore del bit 0, il valore del flag va posto sul bit 2 del dato.

In un trasferimento da memoria a memoria il byte da spostare viene letto dalla locazione sorgente e spostato nel registro temporaneo,**TEMP** e da questo spostato nella locazione di memoria di destinazione. Questo registro può essere solo letto ed ha indirizzo relativo pari a $D.

Il registro **CNTRL** è suddiviso 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

| *bit* | *azione svolta se il bit è posto ad 1* |
|---|---|
| 0 | **TC0: termine conteggio per il canale 0** |
| 1 | **TC1: termine conteggio per il canale 1** |
| 2 | **DREQ0: richiesta inoltrata al canale 0** |
| 3 | **DREQ1: richiesta inoltrata al canale 1** |
| 4 | **non utilizzato** |
| 5 | **abilita trasferimento da memoria a memoria** |
| 6 | **in un trasferimento memoria-memoria, impone che l'indirizzo sorgente deve rimanere costante per trasferire un byte in più locazioni di memoria (inizializzazione di un blocco di memoria a valore costante)** |
| 7 | **abilita il DMA controller** |

**Tabella** Errore. Nel documento non esiste testo dello stile specificato.**-2-Significato dei bit di CNTRL**

| *it* | *azione svolta se il bit è posto ad 1* |
|---|---|
| 0 | **TC0: termine conteggio per il canale 0** |
| 1 | **TC1: termine conteggio per il canale 1** |
| 2 | **DREQ0: richiesta inoltrata al canale 0** |
| 3 | **DREQ1: richiesta inoltrata al canale 1** |
| 4 | **non utilizzato** |
| 5 | **abilita trasferimento da memoria a memoria** |
| 6 | **in un trasferimento memoria-memoria, impone che l'indirizzo sorgente deve rimanere costante per trasferire un byte in più locazioni di memoria (inizializzazione di un blocco di memoria a valore costante)** |
| 7 | **abilita il DMA controller** |

Tabella 4-10-Significato dei bit di CNTRL.

Un quadro riassuntivo sull'indirizzamento dei registri e dei flag del componente è riportato in **Errore. L'origine riferimento non è stata trovata.**tab. 4-11.

| *Indirizzo relativo (Hex)* | *Tipo di accesso* | *Nome del registro o del flag* |
|---|---|---|
| 0 | W/R | **CADDR0** |
| 1 | W/R | **CCOUNT0** |
| 2 | W/R | **CADDR1** |

| | | |
|---|---|---|
| 3 | W/R | **CCOUNT1** |
| 8 | W/R | **CNTRL** |
| 9 | W | **RF0/RF1** |
| A | W | **MF0/MF1** |
| B | W | **MODE0/MODE1** |
| D | R | **TEMP** |

**Tabella** Errore. Nel documento non esiste testo dello stile specificato.**-3 Indirizzamento dei registri e dei flag del DMA controller**

## Descrizione dei comandi

Il dispositivo ha dei comandi attivabili scrivendo all'indirizzo indicato in *Tabella Errore. Nel documento non esiste* **testo dello stile specificato.**-4-Indirizzi dei comandi del DMA controller. I comandi disponibili sono:

- **RESET:** riporta il componente nello stato iniziale, azzerando tutti i registri; per attivarlo basta accedere in scrittura all'indirizzo relativo $D;

- Clear Mask Flag (**CMF**): cancella tutti i flag MF, si attiva scrivendo all'indirizzo relativo $E;

- Write All Mask Flag (**WAMF**): 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 | Tipo di accesso | Nome del comando |
|---|---|---|
| $D | W | **RESET** |
| $E | W | **CMF** |
| $F | W | **WAMF** |

**Tabella** Errore. Nel documento non esiste testo dello stile specificato.**-4-Indirizzi dei comandi del DMA controller**

## Programmazione

Prima che sia effettuata la richiesta al DMA di un trasferimento dati si devono inizializzare i valori nei registri del canale interessato al trasferimento e nel registro di controllo.

Ad esempio, 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.

Si riportano di seguito due esempi di codice assembler che inizializzano il controllore DMA, per un trasferimento di tipo memoria-dispositivo di I/O e memoria-memoria. Si suppone che A0 sia stato precedentemente caricato con l'indirizzo base del DMA controller. Il codice seguente è relativo a un trasferimento di tipo SINGLE dalla memoria all'interfaccia collegata al canale 0. Il numero di byte da trasferire è $20, gli indirizzi sono quelli crescenti a partire da $1000. Il trasferimento impiega 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 sulle stesse posizioni (ciò, ad esempio, ha senso nei casi in cui si debba fare un refresh ciclico di un'area di memoria come quella video).

```
MOVE.W#$1000,0(A0)      //indirizzo inizio blocco sorgente
MOVE.B#$20,1(A0)        //N.ro byte da trasferire
MOVE.B#$10,$B(A0)       //in MODE: 0x0100x0, modo singolo,inc
                        //conteggio,auto inizializzazione, trasferimento
                        //mem->interfaccia, canale 0
MOVE.B#$80,$8(A0)       //in CTRL abilitazione DMA
```

L'operazione di trasferimento inizia quando la periferica collegata al canale 0 spedisce una richiesta al DMA e quest'ultimo l'accetta. La stessa richiesta, invece che mediante un segnale hardware, potrebbe anche realizzarsi in software mediante l'uso del flag RF0 (o RF1 per l'altro canale) con la seguente istruzione:

```
MOVE.b      #$08,9(A0)          //fa uso del flag RF0 per avviare il DMA
```

Il secondo esempio è relativo al trasferimento memoria-memoria di un blocco di byte. In tal caso le operazioni da eseguire sono:

- scrivere all'indirizzo relativo $**0** (registro **CADDR0**) e $**2** (registro **CADDR1**) rispettivamente gli indirizzi della locazione iniziale del blocco in memoria da trasferire e dell'indirizzo di inizio della 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.

Di seguito è riportato l'esempio di inizializzazione relativo a tale tipo di trasferimento di un blocco di 64 byte ($40) a partire dall'indirizzo (sorgente) $1000. Esso deve essere trasferito nella zona di memoria successiva all'indirizzo $2000 (destinazione). Infine, è stata disabilitata l'autoinizializzazione sia per il canale sorgente che per quello destinazione.

```
MOVE.W#$1000,$0(A0)          //inizio blocco sorgente
MOVE.W#$2000,$2(A0)          //inizio blocco destinazione
MOVE.B#$40,$3(A0)            //N.ro byte da trasferire in CCOUNT
MOVE.B#$00,$B(A0)            //in MODE0 modalità di trasferimento
MOVE.B#$01,$B(A0)            //in MODE1 modalità di trasferimento
MOVE.B#$A0,$8(A0)            //in CNTRL abilitazione DMA con
                             //operazione mem-to-mem
```

Quando un device collegato ad un canale del DMA invia una richiesta e il flag MF di quel canale è uguale a zero, viene settato il bit corrispondente in CNTRL (il bit 2 per il canale 0 e il bit 3 per il canale 1). Se il bit 7 del registro CNTRL (che abilita il dispositivo) è uguale a uno, il dispositivo spedisce al processore una richiesta di bus che viene automaticamente concessa.

La direzione del trasferimento dipende dal valore del bit 3 del registro MODE associato al canale; se pari a 0, il trasferimento è dalla memoria al device, altrimenti è nella direzione opposta. Se il bit 5 di CNTRL è uguale ad 1, il trasferimento è, invece, effettuato da memoria a memoria.

Preso il possesso 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. 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.

Completato un ciclo di bus, questo viene rilasciato al processore se il valore del bit 7 del registro MODE è zero (trasferimento a singolo byte); se posto a 1, si susseguono, invece, tutti gli altri cicli necessari al trasferimento dell'intero blocco. Al termine del trasferimento di un blocco, cioè quando il valore **CCOUNT** diventa nullo, sono eseguite le seguenti operazioni:

- 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) a seconda che si sia 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, 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**. Essi,alla scrittura di **CADDR** e **CCOUNT,** assumono il loro stesso valore.

Appendice

<vedere data sheet allegato del dispositivo 82C237 commerciale>