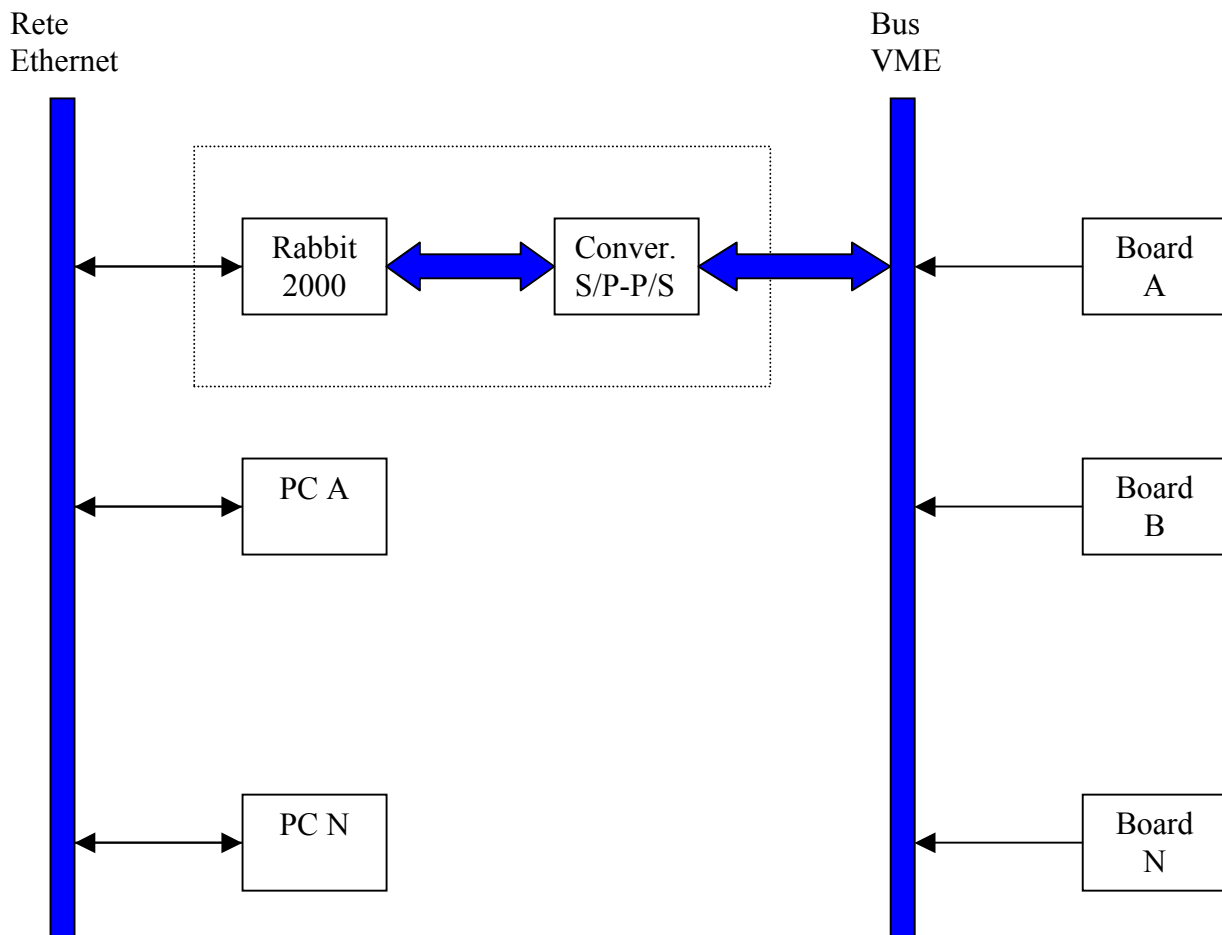


L'obiettivo di questo progetto è la realizzazione di un dispositivo in grado di funzionare da interfaccia tra una rete Ethernet e un bus VME come illustrato nella seguente figura:



Immaginiamo quindi che un utente connesso sulla rete Ethernet abbia l'esigenza di scaricare o di caricare, tramite il protocollo FTP, dati e file su una o più scheda connesse al bus VME.

Chiaramente, poiché questa operazione richiede lo scambio di dati tra due bus completamente differenti, è necessario introdurre un dispositivo in grado di interfacciarli.

Una possibile soluzione sarebbe quella di implementare l'intero protocollo TCP/IP, Ethernet, e infine l'applicazione FTP su un FPGA, quindi in modalità totalmente hardware. In realtà una soluzione di questo tipo risulterebbe particolarmente complessa, essendo legata alla conoscenza approfondita di questi protocolli.

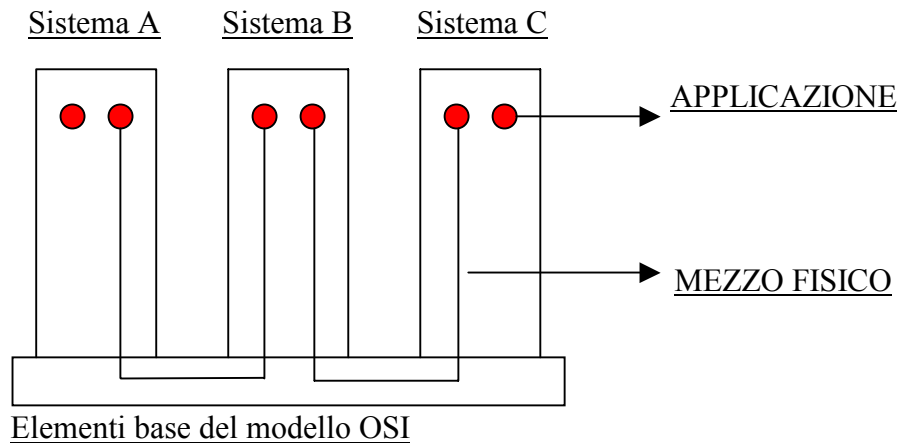
La soluzione più razionale è dunque quella di utilizzare un dispositivo controllato da una CPU.

Il dispositivo in questione è costituito dal μ Processore Rabbit 2000 prodotto da Zworld.

Prima di descrivere il funzionamento del processore, è necessario discutere alcuni argomenti introduttivi sul funzionamento di una rete Ethernet e di un μ P.

Panoramica sul modello ISO/OSI

Il modello ISO/OSI (International Standard Organization/Open Source Interconnection) definisce un'architettura per l'interconnessione di sistemi aperti. Per sistemi intendiamo l'insieme di uno o più elaboratori con il proprio software, le periferiche, i terminali, ecc..



In questo modello assume particolare rilevanza la definizione di “Architettura a livelli”, come di seguito illustrato:

7) APPLICAZIONE
6) PRESENTZIONE
5) SESSIONE
4) TRASPORTO
3) RETE
2) DATA LINK
1) FISICO

Architettura del modello ISO/OSI

Gli elementi fondamentali di questa architettura sono:

- i livelli;
- le entità;
- i punti di accesso al servizio (SAP);
- le connessioni;

I livelli adiacenti comunicano tra di loro per mezzo di interfacce. Ogni livello può avere più entità.

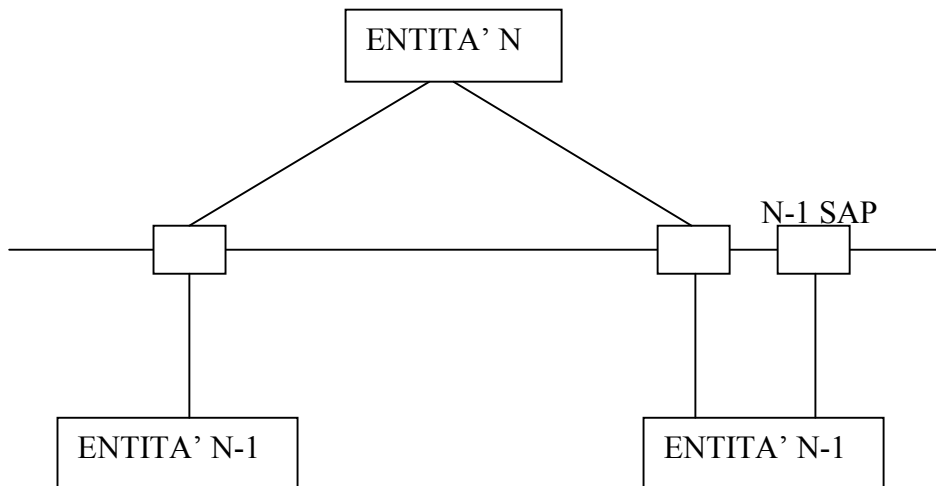
Le entità che appartengono allo stesso livello su sistemi differenti prendono il nome di “peer-entities”.

Un livello N fornisce servizi di livello N alle entità di livello N-1.

Entità di livello N usano servizi di livello N-1 per comunicare.

Le entità di livello 1 usano direttamente i mezzi fisici.

Le entità usano e forniscono servizi per mezzo dei SAP



Due o più entità di uno stesso livello comunicano tra di loro per mezzo di protocolli. Due o più entità di livello N su sistemi diversi possono comunicare tramite protocolli di livello N-1 e la connessione è stabilita tra due SAP di livello N-1.

Vediamo qual' è la funzione dei livelli:

Livello 7 (Applicazione): è quello dei programmi applicativi, attraverso il quale l' utente finale utilizza la rete;

Livello 6 (Presentazione): gestisce la sintassi dell' informazione da trasferire; sono previste tre diverse sintassi: astratta, che rappresenta la definizione formale dei dati da trasferire; concreta locale, ovvero come i dati vengono presentati localmente; di trasferimento, cioè come i dati sono codificati durante il trasferimento.

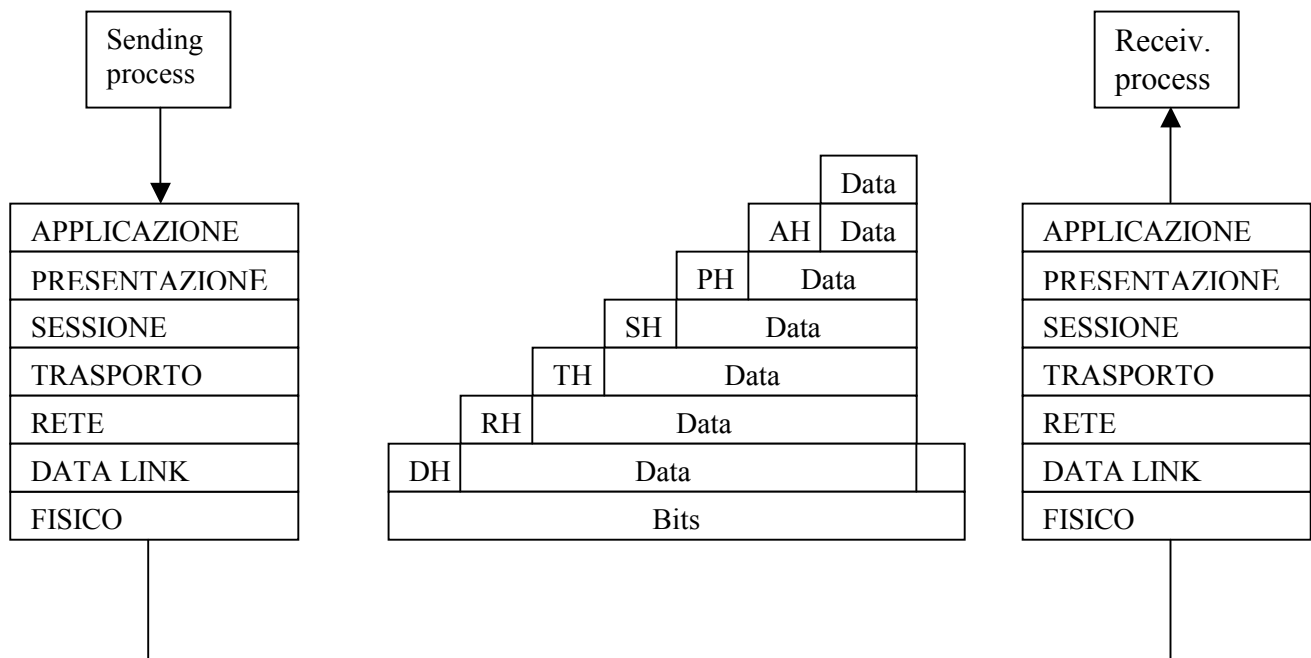
Livello 5 (Sessione): organizza il dialogo tra due applicativi e lo scambio dei dati. Permette di aggiungere servizi più avanzati come la gestione del dialogo (mono o bidirezionale).

Livello 4 (Trasporto): trasferisce in modo trasparente le informazioni tra entità del livello sessione. Questo livello è gestito dal protocollo TCP.

Livello 3 (Network): gestisce l' instradamento dei messaggi e provvede ad instradamenti alternativi in caso di necessità (routing). IP è il protocollo che gestisce questo livello.

Livello 2 (Data link): gestisce la trasmissione affidabile dei frame.

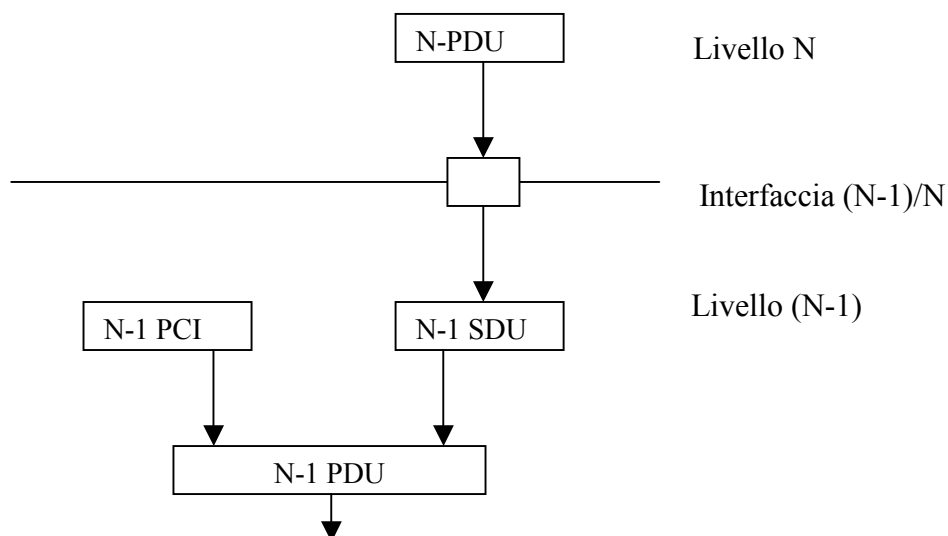
Livello 1 (Fisico): trasmette sequenze binarie sul canale di comunicazione.



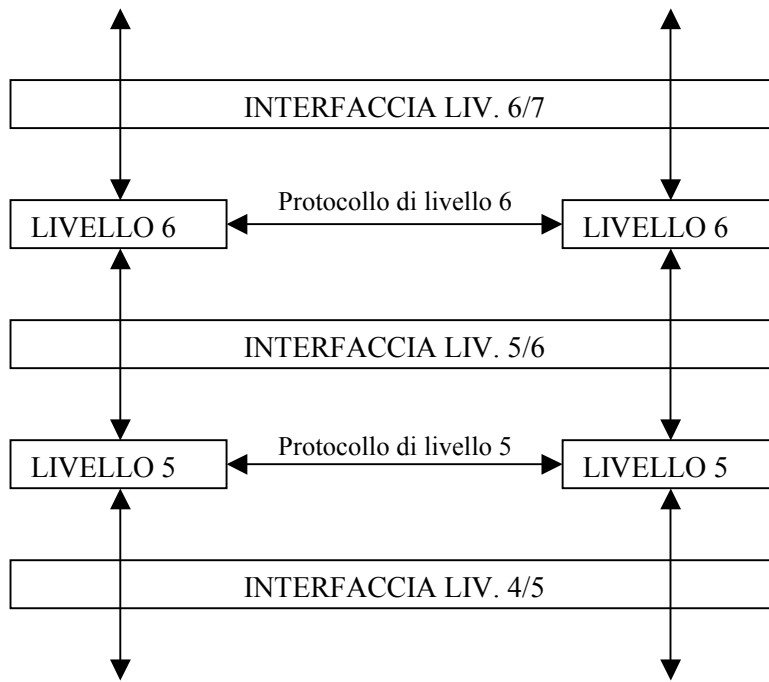
PDU – Protocol Data Unit

Ogni livello N aggiunge ai dati provenienti dal livello superiore alcune informazioni di controllo. Il tutto verrà poi passato al livello inferiore che opera in modo analogo. I dati generati da un protocollo di livello N prendono il nome di N-PDU. Nell' attraversamento dell' interfaccia tra il livello N e il livello N-1 essi diventano una (N-1)-SDU (Service Data Unit).

La (N-1)-PDU si costruisce preponendo alla (N-1)-SDU una (N-1)-PCI (Protocol Control Information) che contiene una informazione di controllo del protocollo.

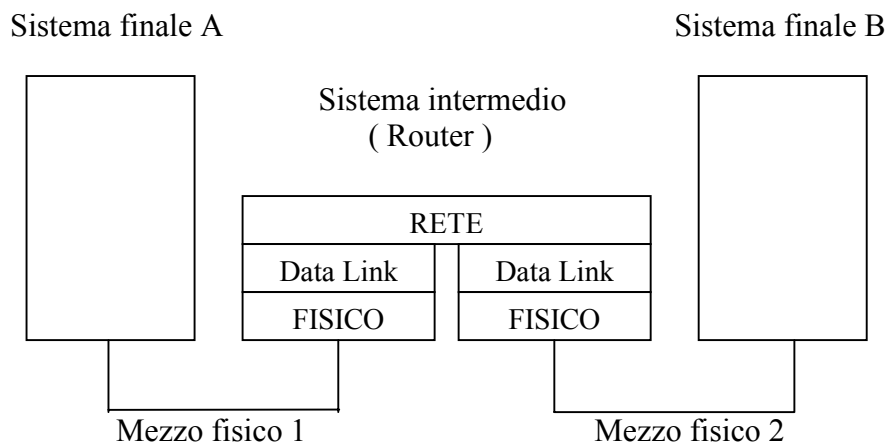


Possiamo generalizzare il modello nel modo seguente:



Anche se esiste un protocollo di livello, due livelli su sistemi diversi comunicano sempre tramite i mezzi fisici.

A volte lo scambio di informazioni tra due sistemi finali non è diretto, ma passa attraverso dei sistemi intermedi, che assumono la funzione di 'relaying', ovvero di inoltratrici di informazioni:

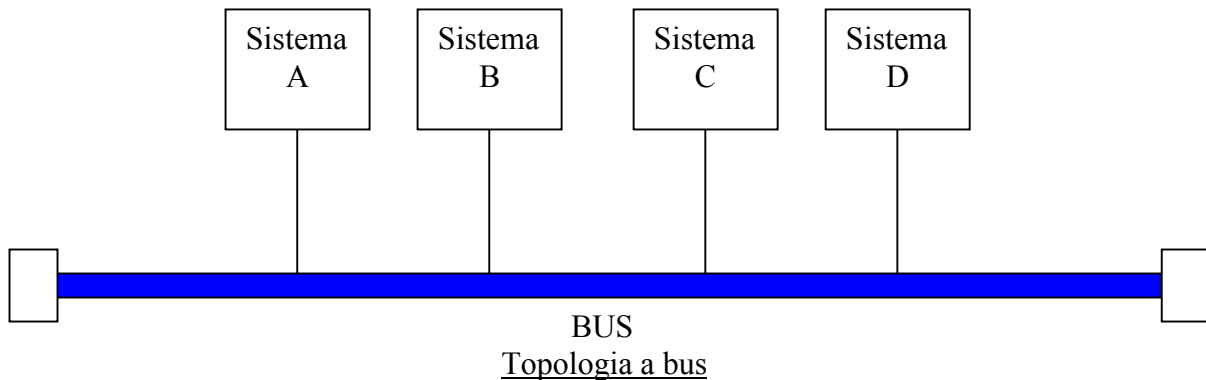


A seconda del livello in cui è posto il sistema intermedio si parla rispettivamente di:

- REPEATER se a livello 1
- BRIDGE se a livello 2
- ROUTER se a livello 3
- GATEWAY se a livello 7

Ethernet

La topologia di una rete IEEE 802.3, nota con il nome Ethernet, è realizzata attorno ad un bus condiviso, come mostrato in figura.



La rete Ethernet opera ad una velocità di 10 Mbit/s. Lo standard IEEE 802.3 specifica solo una parte del modello ISO/OSI, ovvero il livello fisico e il sottolivello MAC del livello data link.

Il protocollo di accesso al canale (MAC)

La rete Ethernet utilizza come sistema di accesso multiplo (o MAC – Medium Access Control) il protocollo CSMA/CD (Carrier Sense Multiple Access/Collision Detection), in base al quale una stazione che deve inviare un pacchetto di dati, ascolta il bus: se esso è libero, la stazione può iniziare a trasmettere. Nonostante il meccanismo di ‘carrier sense’, è possibile che due stazioni interferiscano tra loro generando una collisione. Questo inconveniente è determinato dal fatto che il tempo di propagazione del segnale sul bus non è nullo, per cui una stazione A non può sentire l’occupazione del canale da parte di una stazione B fino a quando il segnale generato da B non arriva ad A.

Per rilevare e verificarsi di queste collisioni, una stazione in fase di trasmissione continua ad ascoltare i segnali sul bus, confrontandoli con quelli da essa generati. Nel caso in cui sia rilevata una collisione vengono effettuate le seguenti azioni:

- la stazione trasmittente interrompe la trasmissione e invia una sequenza detta di ‘jamming’, composta da 32 bit per avvertire le altre stazioni della collisione;
- le stazioni in ascolto intercettano la sequenza e scartano i bit ricevuti;
- la stazione trasmittente ripete il tentativo di trasmissione dopo un tempo generato in modo casuale. Il numero massimo di tentativi di ritrasmissione è 16.

Il frame della rete Ethernet presenta il seguente formato:

Preamble	Start frame	Indirizzo destinat.	Indirizzo sorgente	Lunghezza dati	Dati	Pad	Check
----------	-------------	---------------------	--------------------	----------------	------	-----	-------

I campi hanno il seguente significato:

- Preamble: formato da 7 byte uguali a 10101010. Questo produce, a 10 Mbit/sec., un’ onda quadra a 10 Mhz per 5,6 us che sincronizza il clock del ricevitore con quello del trasmettitore.
- Un byte delimitatore uguale a 10101011.

- Indirizzo a 6 byte. Si può specificare un singolo destinatario, un gruppo di destinatari (multicast) oppure tutte le stazioni (l'indirizzo è costituito da una serie di uni).
- Lunghezza dati: indica quanti byte ci sono nel campo dati (da 0 a 1500).
- Dati: contiene i dati provenienti dal livello superiore.
- Pad: se il frame è più corto di 64 byte (escludendo preamble e delimiter) viene portato a tale lunghezza in questo campo.
- Check: contiene un codice di controllo.

Il frame indicato presenta le seguenti caratteristiche generali:

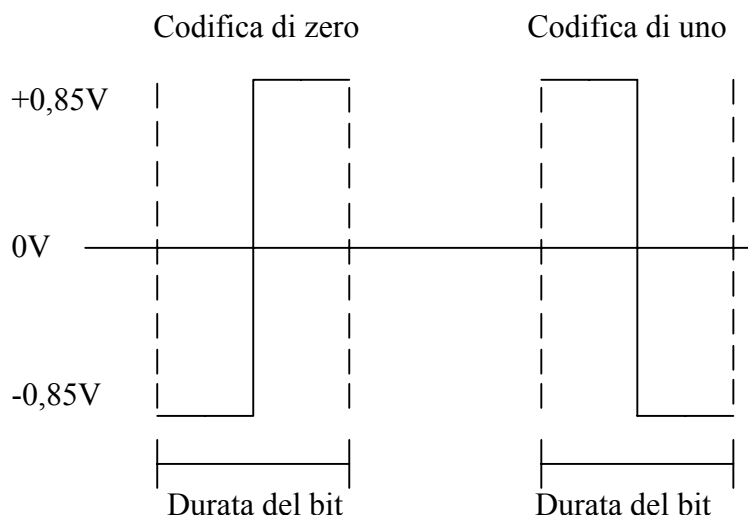
- lunghezza minima di 64 byte
- lunghezza massima di 1518 byte

Questa limitazione è dovuta al fatto che, indicando con T il tempo di propagazione del segnale tra due stazioni qualsiasi presenti sulla rete, se la stazione A inizia a trasmettere all'istante t , la stazione B non si accorge che il canale è occupato dalla stazione A fino all'istante $T+t$, e quindi prima di tale istante può inviare un proprio messaggio. All'istante $T+t$ la stazione B rileva la collisione, interrompe la trasmissione e invia il pacchetto di jamming. La stazione A riceve il pacchetto all'istante $t+2T$ e soltanto a questo istante, cioè dopo un intervallo $2T$, è in grado di rilevare la sovrapposizione del proprio messaggio con quello generato da B.

Se indichiamo con T il tempo massimo di propagazione tra due qualsiasi stazioni presenti sulla rete, $2T$ indica il tempo massimo di trasmissione per una stazione prima di accorgersi di una collisione. Il tempo $2T$ prende il nome di 'round-trip delay'. La durata di un qualsiasi messaggio deve essere superiore di $2T$ in modo che la stazione che trasmette sia in grado di rilevare una sua eventuale sovrapposizione con altri messaggi. Questa condizione impone una limitazione sulla minima lunghezza consentita per un messaggio.

Per esempio, nel caso di una rete Ethernet, è prevista una lunghezza massima di 2,5 Km con un round-trip delay di 57,6 μ s. Infatti questo è il tempo necessario per trasmettere 72 byte, di cui 8 sono formati da preamble e delimiter e 64 dal resto del frame.

In Ethernet si utilizza per i dati la codifica Manchester che prevede una transizione del segnale nel mezzo di ogni bit, uno o zero che sia.



Codifica Manchester

Un problema legato a questo tipo di codifica sta nel fatto che, a parità di velocità di trasmissione, è richiesta una banda doppia rispetto alla codifica diretta, poiché ogni bit richiede la trasmissione di due valori distinti.

L' IP (Internet Protocol)

Il protocollo IP è quello che si occupa di gestire il livello network. In particolare esso esegue i seguenti passi:

in trasmissione:

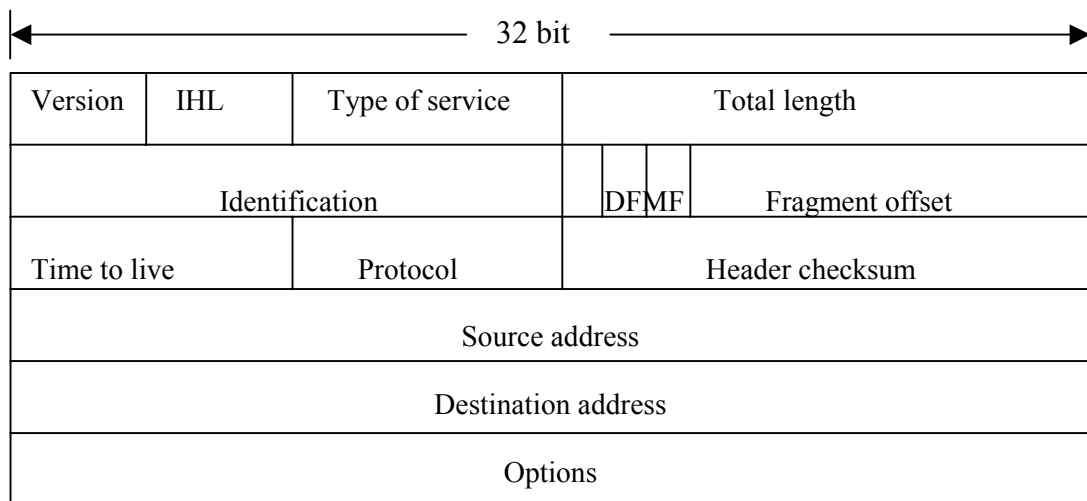
- riceve i dati dal livello trasporto convertendoli in pacchetti di dimensione massima pari a 64Kbyte;
- instrada i pacchetti sulla subnet eventualmente frammentandoli;

in ricezione:

- riassembla gli eventuali frammenti in pacchetti;
- estrae da questi i dati del livello trasporto;
- consegna al livello trasporto i dati nell'ordine in cui sono arrivati;

Un pacchetto IP è formato da un header e da un campo dati.

L' header ha una parte fissa di 20 byte e una parte opzionale di lunghezza variabile.



Formato dell' header IP

I campi hanno il seguente significato:

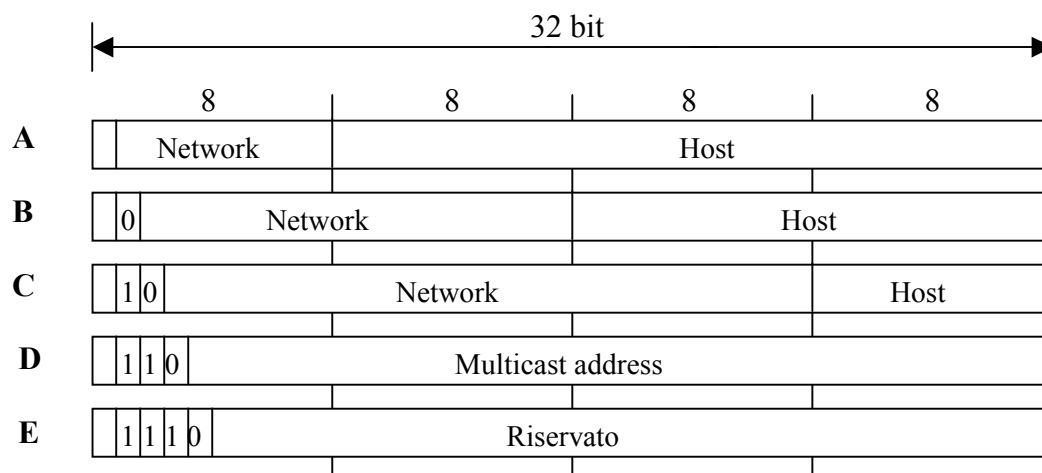
- Version: indica la versione del protocollo (attualmente 4);
- IHL: lunghezza dell' header in parole da 32 bit (minimo 5, max 15);
- Type of service: contiene informazioni sulla velocità e l' affidabilità;
- Total length: lunghezza totale del pacchetto (max. 65535 byte);
- Identification: identifica i frammenti di uno stesso pacchetto;
- DF: se posto uguale ad 1 non si deve frammentare il pacchetto;
- MF: se uguale ad 1 il pacchetto non è ancora finito;
- Fragment offset: indice del frammento nel pacchetto;
- Time to live: è un contatore che partendo da 255, viene decrementato ogni secondo. Quando arriva a zero, il pacchetto viene scartato;
- Protocol: codice del protocollo di livello trasporto a cui consegnare il pacchetto;
- Header checksum: campo di controllo del solo header;
- Source address: indirizzo del mittente;
- Destination address: indirizzo del destinatario;

Un indirizzo IP è formato da 32 bit e codifica:

- **network number**, cioè il numero assegnato alla rete IP (detta **network**) su cui si trova l'elaboratore. Una network è caratterizzata dal fatto di essere costituita da un unico canale di comunicazione cui sono connessi tutti gli host della network stessa;

- **host number**, cioè il numero assegnato all'elaboratore.

I formati ammessi per gli indirizzi sono i seguenti:



Pertanto possono esistere:

- 126 network di classe A, le quali possono contenere 16 milioni di host ciascuna;
- 16382 network di classe B, con circa 64.000 host ciascuna;
- 2 milioni di network di classe C, con 254 host ciascuna.

Gli indirizzi sono usualmente espressi nella **dotted decimal notation**, cioè i valori dei singoli byte sono espressi in decimale e sono separati da un punto, come nell'indirizzo: 141.192.140.37..

In tale notazione, è possibile rappresentare separatamente il network number e l'host number. Per distinguerli, il primo è seguito da un punto. Ad esempio, nel caso dell'indirizzo IP precedente (che è relativo ad una network di tipo B), si ha:

il network number è 141.192.;

l'host number è 140.37;

Il TCP (Transmission Control Protocol)

Il TCP gestisce il livello trasporto e in particolare si occupa di:
in trasmissione:

- accetta i dati dal livello superiore, cioè applicazione;
- spezza i dati in segmenti;
- li consegna al livello network;

in ricezione:

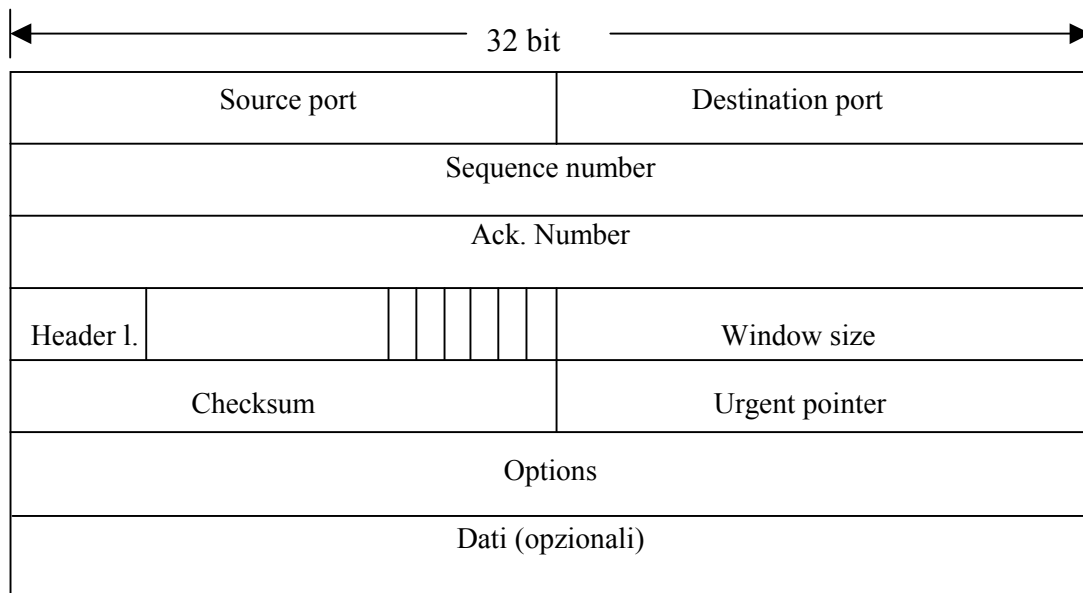
- riceve segmenti dal livello network;
- li rimette in ordine, eliminando eventuali errori;
- consegna i dati ordinati al livello applicazione;

I servizi di TCP si ottengono creando connessione di livello trasporto tra una coppia di punti d'accesso che prendono il nome di **socket**, caratterizzato da un **socket number** formato dalla coppia: IP-address: Port number. I port number sono formati da 16 bit. Ad ogni port number è associato un servizio di livello TCP:

Port number	Servizio
7	Echo
20	FTP (control)
21	FTP (data)
23	Telnet
25	SMTP
80	HTTP
110	POP3

L header del protocollo TCP è formato da:

- una parte fissa da 20 byte;
- una parte opzionale;



Formato dell' header TCP

I campi hanno il seguente significato:

- Source e destination port: indicano i punti finali tra cui avviene la connessione ;
- Sequence number: indica il numero d' ordine del primo byte nel campo dati;
- Ack number: il numero d' ordine del successivo byte atteso;
- Header l.: quante parole da 32 bit ci sono nell' header;
- URG: 1 se urgent pointer è usato, 0 altrimenti;
- ACK: 1 se l' ack number è valido;
- PSH: dati da consegnare senza attendere il riempimento del buffer;
- RST: reset della connessione;
- SYN: richiesto in fase di setup della connessione;
- FIN: usato per rilasciare una connessione;
- Window size: window size dice quanti byte possono essere spediti a partire da quello (compreso) che viene confermato con l'ack number.
- Checksum: simile a quello di IP;
- Urgent pointer: puntatore ai dati urgenti;
- Options: funzioni opzionali.

Architettura di un μP

Il μ Processore rappresenta il primo e fondamentale elemento di un sistema a microcomputer. Esso è costituito, in generale, dai seguenti elementi:

- l' accumulatore: è un registro che contiene un dato nelle operazioni aritmetiche, logiche e di I/O. I dati possono essere letti dalla memoria nell' accumulatore e viceversa. Le operazioni logiche e aritmetiche coinvolgono due operandi, uno posto nell' accumulatore e il secondo prelevato dalla memoria. Il risultato dell' operazione è ancora posto nell' accumulatore. Inoltre serve da registro nelle operazioni di I/O.
- i registri di scratch-pad: contengono risultati intermedi dei calcoli e indirizzi.
- I flag: indicano il risultato di un' operazione e lo stato del sistema.
- L' unità aritmetico logica: realizza le operazioni aritmetiche e logiche sui dati provenienti dall' accumulatore e dalla memoria.
- Il program-counter: contiene l' indirizzo della posizione di memoria in cui si trova la successiva istruzione di programma da eseguire.
- Il registro istruzioni: contiene il codice operativo dell' istruzione in corso di esecuzione.
- Il decodificatore di istruzioni e l' unità di temporizzazione e controllo: interpreta il codice operativo necessario a definire il valore dei segnali di controllo necessari all' esecuzione dell' istruzione.
- Buffer per i dati e gli indirizzi: memorizzano dati in ingresso e pilotano buffer in uscita.

Le operazioni in cui si realizza un' istruzione sono:

- ciclo di fetch (acquisizione dell' istruzione): il PC (program-counter) invia alla memoria l' indirizzo della prossima istruzione; in seguito la memoria restituisce il codice operativo al registro istruzioni posto nella CPU:
- ciclo di execute (esecuzione dell' istruzione): il PC viene incrementato in modo da prepararlo al prossimo ciclo di fetch. Il codice operativo viene interpretato dal decodificatore e viene quindi eseguita l' istruzione attraverso la generazione di una opportuna serie di operazioni da parte dell' unità di temporizzazione e controllo.

Vediamo, con l' ausilio della figura seguente, come avvengono queste operazioni.

Nel primo passo del ciclo di fetch l' indirizzo di memoria dell' istruzione da eseguire, contenuto nel program counter, viene inviato alla memoria attraverso l' address bus.

Nel secondo passo la memoria preleva l' istruzione posta nell' indirizzo indicato e la invia al microprocessore sul data bus. Quindi il microprocessore memorizza l' istruzione nel registro istruzioni.

Il terzo passo fa parte del ciclo di esecuzione . L' istruzione viene decodificata, ovvero l' unità di controllo determina quale operazione è stata indicata dall' istruzione.

Contemporaneamente l' indirizzo contenuto nel program counter viene incrementato per renderlo atto a individuare il successivo indirizzo in memoria e quindi prepararlo per il successivo ciclo di esecuzione dell' istruzione.

Infine, nel quarto passo, l' unità di controllo genera gli opportuni segnali di controllo per far sì che l' istruzione precedentemente decodificata venga eseguita. L' esecuzione si traduce in movimenti di dati e in elaborazioni come indicato dall' istruzione.

I segnali di controllo aprono le opportune gate e temporizzano la memorizzazione dei dati negli opportuni registri per realizzare il compito predetto.

Un altro elemento fondamentale di un sistema a μ Computer è costituito dalla memoria, unità in cui vengono memorizzati dati e programmi.

In genere, un microcomputer, è un sistema atto a eseguire un determinato programma. Quest' ultimo non varia frequentemente, al contrario di quanto succede invece con i dati da elaborare.

E' quindi utile distinguere da memoria programma, il cui contenuto cambia molto raramente, dalla memoria dati, i cui contenuti vengono continuamente alterati.

Le operazioni che implicano il movimento dei dati richiedono che venga specificata la posizione da cui prelevare i dati e quella in cui depositarli. Esistono diversi metodi di indirizzamento dei dati, ciascuno con caratteristiche diverse, ma tutti ugualmente importanti.

Ogni istruzione può considerarsi come formata da due parti:

- codice operativo;
- campo dati;

A sua volta il campo dati può essere:

- un operando;
- un indirizzo di una posizione di memoria;
- un indirizzo di un registro;

Gli organi di CPU che sono coinvolti nei vari metodi di indirizzamento sono:

- l' accumulatore;
- il registro indice;
- i registri dello stack;

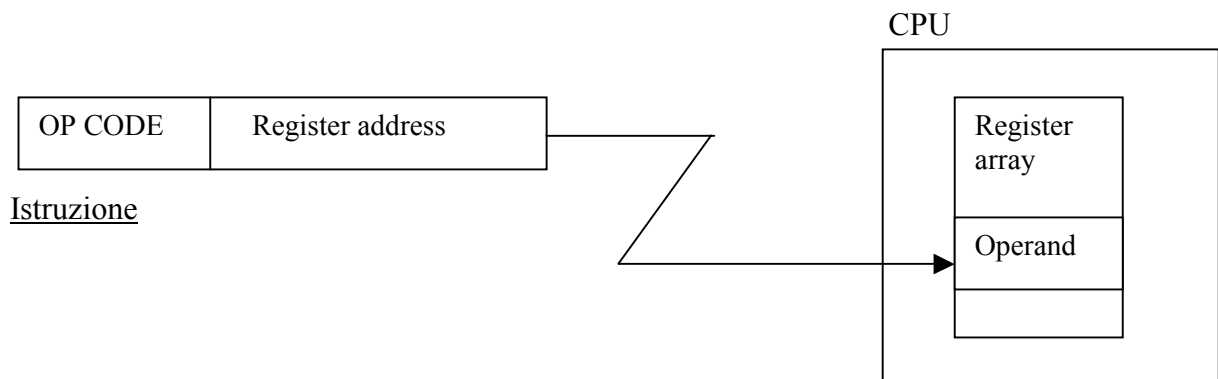
Vediamo in dettaglio quali sono i metodi di indirizzamento. Iniziamo con il metodo che prevede che l' operando sia contenuto nella stessa istruzione, ovvero il metodo di **indirizzamento immediato**.



Istruzione

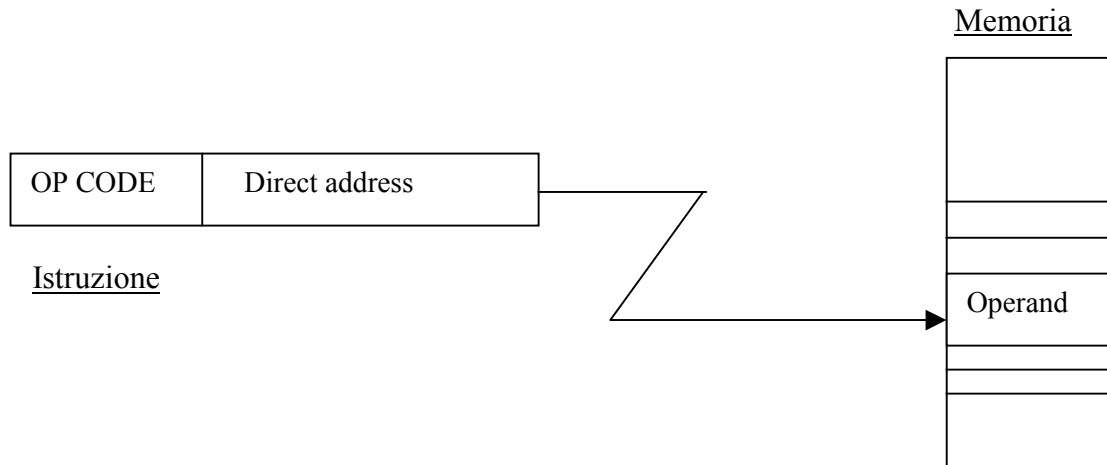
In esso, l' operando è contenuto nell' istruzione di memoria o la segue. E' un metodo utile per caricare delle costanti, ovvero valori dell' operando definiti dal programmatore.

Un metodo di indirizzamento in cui invece l' operando è contenuto in un registro è il metodo di **indirizzamento di un registro**.

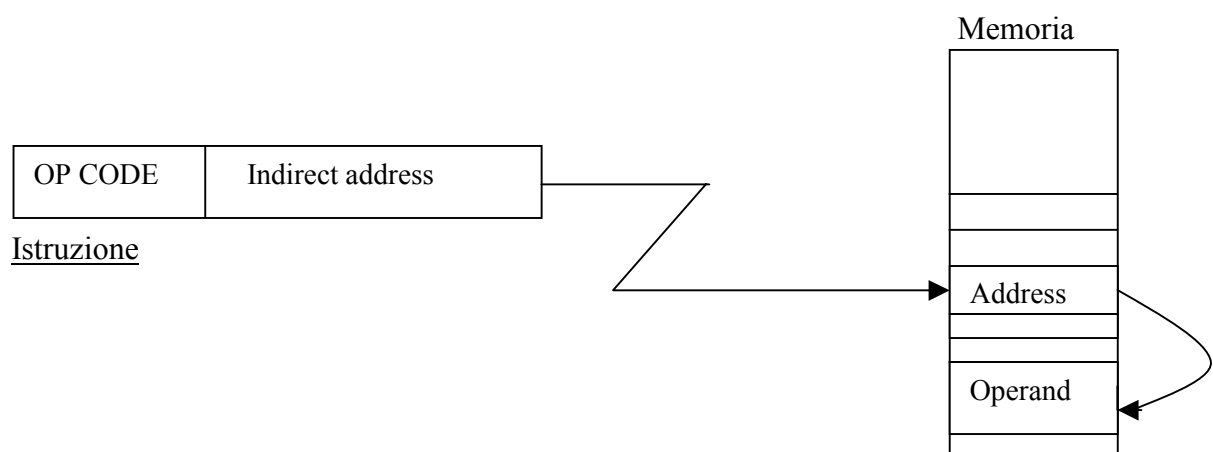


Nei metodi di indirizzamento seguenti, invece, l'operando è contenuto in una posizione di memoria:

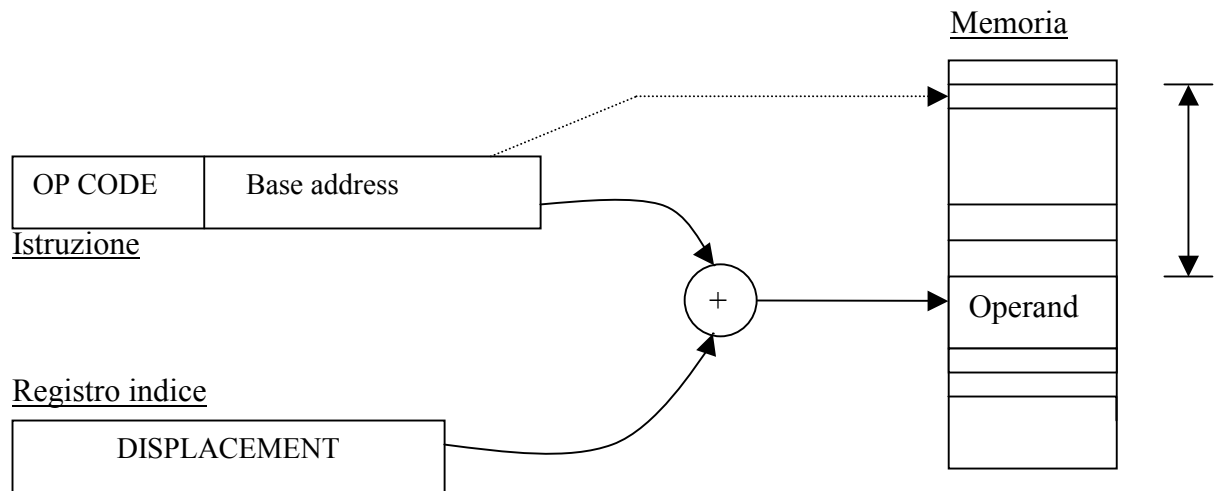
-indirizzamento diretto: nell'istruzione è contenuto l'indirizzo della posizione di memoria in cui è posto l'operando.



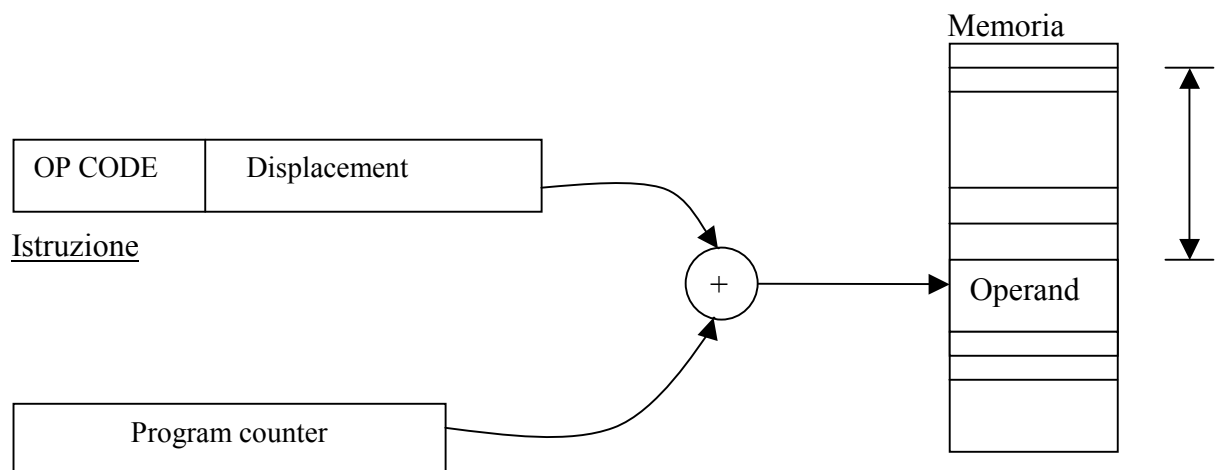
-indirizzamento indiretto: l'istruzione contiene l'indirizzo di una posizione di memoria, la quale, a sua volta, contiene l'indirizzo dell'operando o l'indirizzo di un altro indirizzo (indirizzamento a più livelli).



-indirizzamento con registro indice: l' indirizzo dell' operando si ottiene sommando il contenuto del registro indice (displacement) all' indirizzo base contenuto nell' istruzione

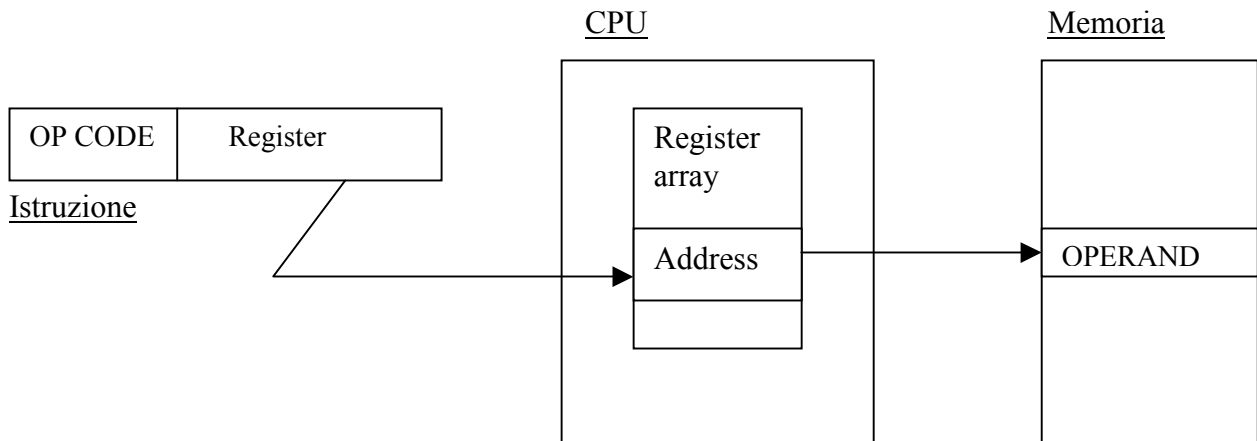


-indirizzamento relativo: l' indirizzo dell' operando è ottenuto sommando i contenuti del program counter ad un valore costante (displacement) contenuto nell' istruzione.

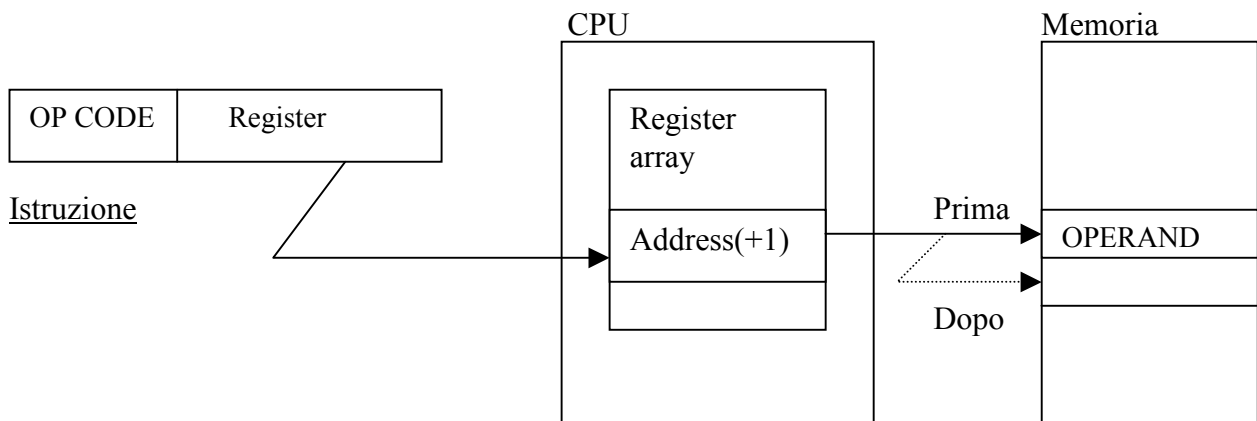


Esistono, infine, alcuni metodi di indirizzamenti che si ottengono dalla composizione dei precedenti.

-indiretto da registro: l'indirizzo dell'operando è contenuto nel registro di CPU che viene specificato in modo implicito o esplicito nell'istruzione.



-indiretto da registro con incremento o decremento automatico: questo tipo di indirizzamento è simile al precedente. La differenza sta nel fatto che il registro viene incrementato o decrementato automaticamente dopo o prima del prelievo (fetch) dell'operando. Viene usato per accedere a dati memorizzati in maniera sequenziale come **STACK** (LIFO-Last In First Out) e **QUEUE** (FIFO-First In First Out).



Dunque l'insieme delle istruzioni appena descritte costituisce il programma e attraverso questo è possibile realizzare molte funzioni logiche sequenziali e combinatorie.

Le istruzioni possono essere classificate nel seguente modo:

- istruzioni per il caricamento dei dati nei registri;
- istruzioni per il trasferimento in memoria dei dati presenti nei registri;
- istruzioni di operazioni su dati presenti nei registri;
- istruzioni di operazioni sui dati presenti in memoria;
- istruzioni per il controllo della sequenza di programma;
- istruzioni di I/O;

Abbiamo già visto che ogni istruzione è fondamentalmente costituita dai seguenti campi (che non sono necessariamente presenti contemporaneamente):

- codice operativo;
- identificatore di registro;
- indirizzo di memoria;
- codici di controllo;
- codici di condizione;

Esaminiamo la struttura delle varie classi di istruzioni e le operazioni che esse svolgono.

Istruzioni di caricamento di un registro da memoria (Load)

Istruzioni di caricamento della memoria da un registro (Store)

Queste due classi di istruzioni realizzano scambio di dati tra la memoria e un registro che molto spesso è l'accumulatore. Solitamente il contenuto della posizione di memoria o del registro specificato non viene alterato.

La struttura tipica per queste due classi di istruzioni è molto simile e pongono quindi gli stessi problemi di indirizzamento.

OP-CODE	MODE	ADDRESS
---------	------	---------

L'OP-CODE indica il tipo di operazione che si vuole eseguire (Load o Store).

Il campo MODE permette di definire come deve essere individuato l'indirizzo effettivo della posizione di memoria interessata o il registro su cui si deve operare, secondo quanto visto precedentemente sui metodi di indirizzamento.

Il campo ADDRESS permette di ottenere, in funzione del metodo definito nel campo MODE, il valore dell'indirizzo effettivo di memoria su cui operare.

Istruzioni di operazioni tra registri

Questa classe di istruzioni permette di realizzare operazioni di spostamento (MOVE) di data tra:

- un registro e un altro;
- tra un registro e la memoria di scratch-pad;
- nell'ambito dello stack.

La struttura dell'istruzione è la seguente:

OP-CODE	SOURCE REG.	DEST. REG.
---------	-------------	------------

Dove l' OP-CODE in questo caso rappresenta l' operazione di spostamento e SOURCE e DESTINATION ADDRESS indicano rispettivamente l' indirizzo dei registri tra cui deve avvenire lo scambio.

Operazioni di tipo aritmetico-logico

Vengono generalmente usate per realizzare le operazioni aritmetiche di base, quindi somma e sottrazione, e quelle logiche, cioè AND, OR, ecc. Questa istruzione avviene generalmente tra due parole poste in due registri che solitamente sono l' accumulatore e un secondo registro. Il risultato è generalmente posto nell' accumulatore e vengono inoltre settati alcuni flag che consentano di controllare il risultato dell' operazione e quindi del sistema. L' istruzione ha il seguente formato:

OP-CODE	MODE	ADDRESS
---------	------	---------

Operazioni di controllo della sequenza di programma

Istruzione di salto incondizionato

Con questa istruzione viene caricato nel Program counter l' indirizzo specificato nell' istruzione stessa.

Istruzione di salto condizionato

Permette di controllare una condizione logica specificata nell' istruzione. Nel caso in cui la condizione sia vera si salta all' indirizzo indicato nell' istruzione, altrimenti si continua nella sequenza di programma.

Nel caso di salto condizionato, la condizione di salto è indicata nell' OP-CODE. Può succedere che l' istruzione occupi due parole. In questo caso il primo byte conterrà i campi OP-CODE e MODE e il secondo byte conterrà il campo ADDRESS.

Istruzione di salto a sub-routine

Una sub-routine può considerarsi come una sequenza di codici che possono essere usati in diversi punti di un programma. La sua utilità consiste nel ridurre il tempo di programmazione e la quantità di memoria richiesta per il programma. Lo svantaggio invece è un maggior tempo di esecuzione dovuto alla presenza di istruzioni di richiamo delle sub-routine e di ritorno da esse. Inoltre a ogni richiamo di una sub routine, bisogna salvare lo stato del sistema nello stack e richiamarlo al ritorno, operazione che rallenta notevolmente l' esecuzione del programma.

L'istruzione di salto a sub-routine è indicata nell' OP-CODE.

La struttura delle istruzioni appena descritte è la seguente:

OP-CODE	MODE	ADDRESS
---------	------	---------

Istruzioni di I/O

Sono comandi il cui scopo è di leggere dei dati da un dispositivo esterno o di trasferirli in un dispositivo esterno. L' istruzione deve indicare, attraverso un indirizzo, il dispositivo esterno a cui si rivolge l' operazione. Il formato tipico è il seguente:

OP-CODE	MODE	ADDRESS
---------	------	---------

Abbiamo visto come avviene un'istruzione di salto a sub routine. Pertanto descriviamo cosa avviene durante una chiamata a sub-routine e durante un ritorno da sub-routine.

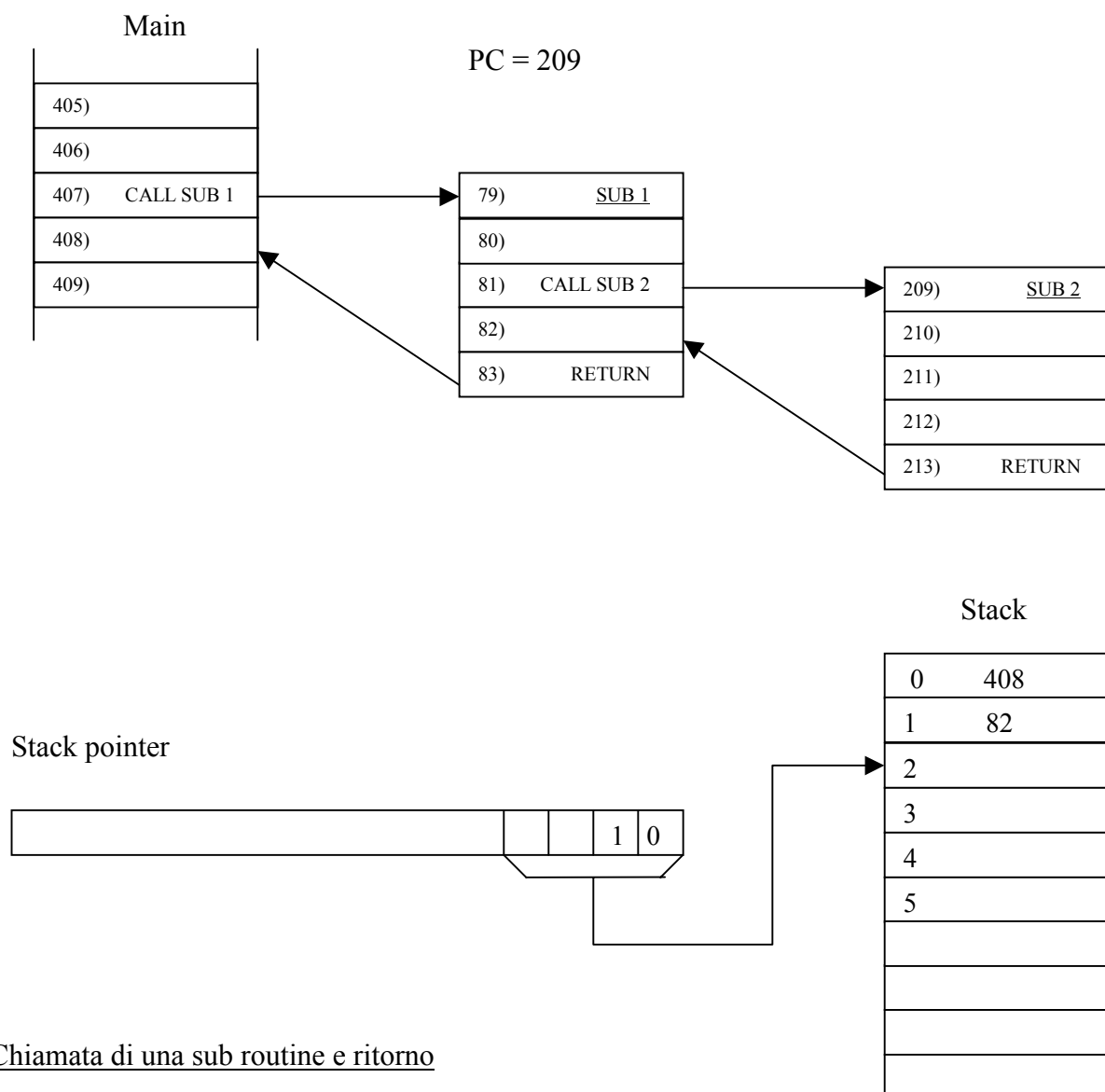
La prima fase di una chiamata a sub-routine si realizza con un incremento del program-counter il quale viene memorizzato in un registro o in una opportuna posizione di memoria. Quindi viene calcolato l'indirizzo effettivo di inizio della sub-routine e lo si carica nel program-counter. Il programma procede eseguendo la sub routine. Alla fine, in presenza dell'istruzione di return, il program counter riprende il valore che aveva precedentemente salvato, puntando alla successiva istruzione nel programma principale.

E' chiaro che durante la prima fase della sub-routine devono essere salvate tutte le informazioni contenute negli organi della CPU, che potrebbero andare distrutte durante l'esecuzione della sub-routine, e che dovranno essere utilizzate al ritorno nel programma principale.

L'area di memoria utilizzata per il salvataggio del program-counter e della CPU prende il nome di stack, a cui è associato un registro, lo stack pointer che contiene l'indirizzo della prima posizione di memoria libera dello stack.

Un'istruzione di richiamo a sub-routine (Call) memorizza il program-counter nello stack ed aggiorna lo stack pointer. L'istruzione di ritorno da sub-routine (Return) produce l'operazione inversa, ovvero ripristino del program-counter e decremento dello stack pointer.

La figura seguente descrive come avvengono queste operazioni.

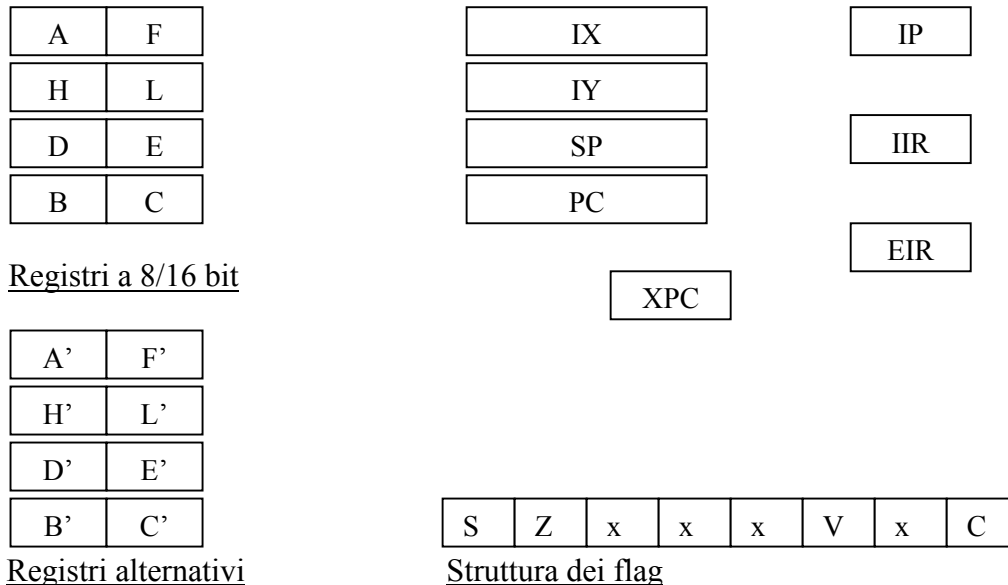


Ad ogni istruzione di CALL, lo stack tende a riempirsi (PUSH), mentre ad ogni istruzione di RETURN, esso tende a svuotarsi (POP). Ad ognuna di queste operazioni, lo stack pointer aggiorna il proprio valore per indicare l'indirizzo della prima area libera dello stack, mentre il program counter viene incrementato o decrementato per indicare l'indirizzo della successiva istruzione da eseguire.

μP Rabbit 2000

Registri

Sono strutturati secondo la figura seguente:



Il registro EIR (richiesta esterna di interrupt) è utilizzato per puntare alla tabella dei vettori di interrupt per gli interrupt generati esternamente. IIR invece è utilizzato per gli interrupt interni.

Il μP è composto da due accumulatori:

- il registro A è un accumulatore a 8 bit per operazioni a 8 bit come ADD e AND;
- HL è un accumulatore a 16 bit per operazioni come ADD HL, DE che somma il contenuto (16 bit) del registro DE con il contenuto dell' accumulatore.

F è il registro di flag o registro di stato, che contiene un valore che dà informazioni sullo stato dell' ultima operazione eseguita. Si tratta di un registro al quale non è possibile accedere direttamente se non con i comandi PUSH e POP.

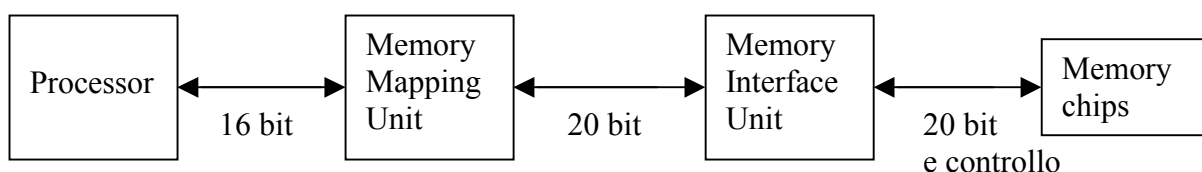
IX, IY e HL possono essere usati anche come registri indice. Essi puntano all' indirizzo di memoria della successiva istruzione da eseguire. Il Rabbit 2000 è in grado di indirizzare un Mbyte o più di memoria, mentre il registro indice può indirizzare direttamente 64k di memoria. Il registro SP è lo stack pointer, usato per l' esecuzione delle subroutine.

IP è il registro che contiene le priorità degli interrupt. E' formato da quattro campi da due bit che contengono informazioni sulle priorità di interrupt.

Mappatura di memoria

Il Rabbit è in grado di indirizzare direttamente 64k di spazio di memoria dati. Il campo indirizzi all' interno delle istruzioni è di 16 bit e anche i registri usati come puntatori agli indirizzi di memoria, il program counter e lo stack pointer sono di 16 bit.

L' MMU (Memory Mapping Unit) riceve indirizzi a 16 bit in ingresso, provenienti dal microprocessore, e porta in uscita indirizzi a 20 bit.



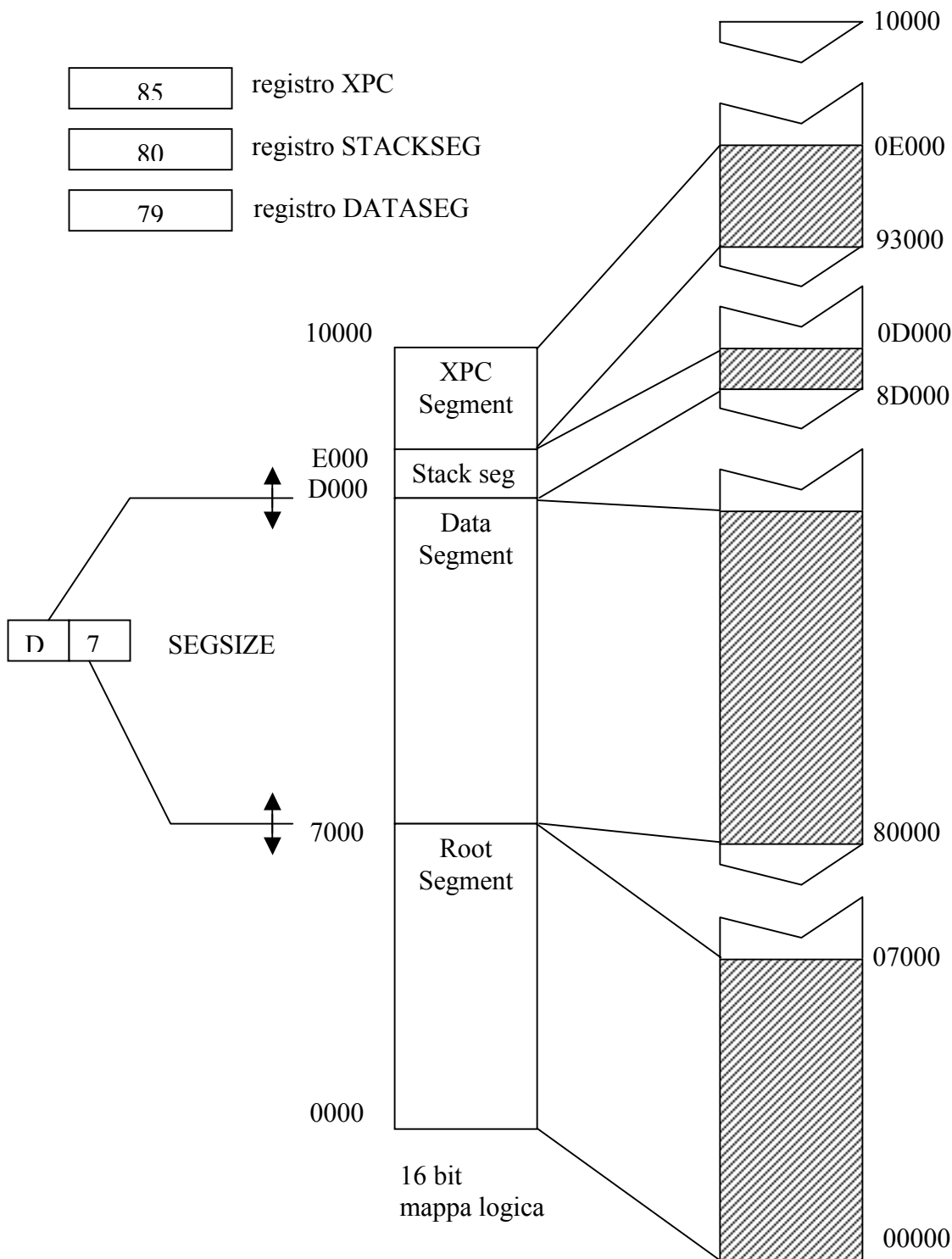
Componenti dell' indirizzamento

Tre registri a segmento vengono usati per mappare i 16 bit di spazio in ingresso in 20 bit di spazio in uscita.

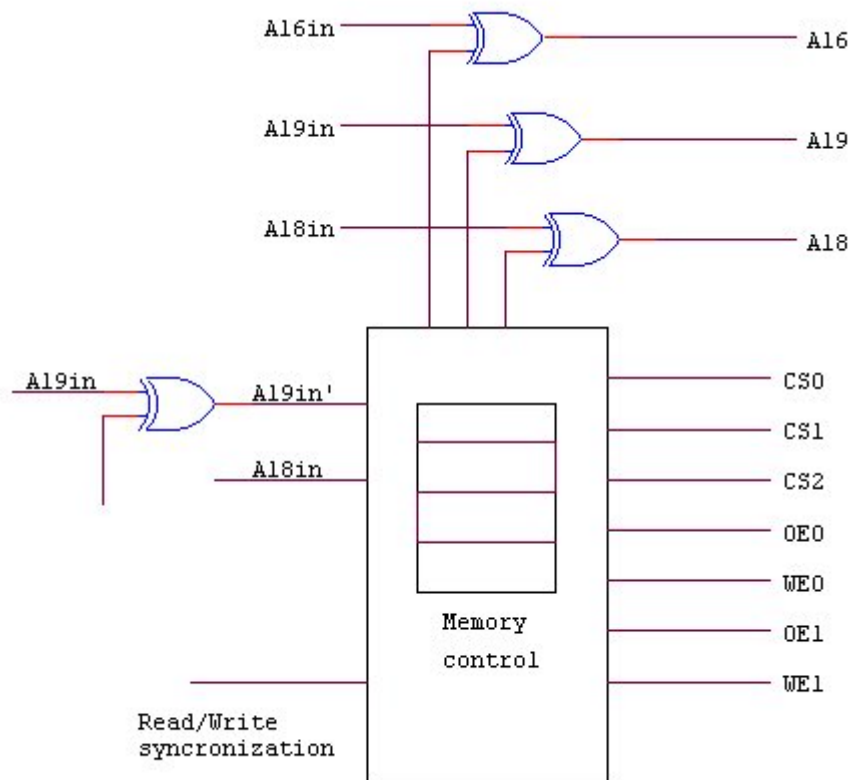
Lo spazio a 16 bit è diviso in 4 zone separate, ciascuna delle quali, tranne la prima o zona root, ha un registro a segmento che viene sommato con l'indirizzo a 16 bit per generare indirizzi a 20 bit. Il registro ha 8 bit che vengono sommati con i 4 bit superiori dell'indirizzo a 16 bit per generare un indirizzo a 20 bit. Ciascuna zona della memoria a 16 bit diventa una finestra per un segmento di memoria nello spazio indirizzo a 20 bit. Il registro SEGSIZE controlla la misura relativa dei quattro segmenti nello spazio a 16 bit ed è formato da un registro a 8 bit che contiene due registri da 4 bit. Il segmento ROOT è mappato alla base della memoria flash e contiene il codice di avvio o altro codice. L'uso del segmento DATI dipende da come è stata settata la memoria. Esso può essere un'estensione del segmento ROOT o contenere dati variabili.

Il segmento STACK è generalmente di 4k e serve per contenere lo stack di sistema. Il segmento XPC è usato normalmente per eseguire codice che non viene salvato nel segmento root o dati.

Vediamo un esempio di come può essere mappata la memoria.



L' Unità Interfaccia di Memoria riceve indirizzi a 20 bit. A seconda della condizione, si possono modificare le linee di indirizzo A16, A18 e A19. Le altre linee passano senza alcuna condizione. Essa gestisce i segnali di controllo per la memoria esterna, ovvero chip select (CS0, CS1 e CS2), write enable (WE0 e WE1) e output enable (OE0 e OE1). I 20 bit di spazio indirizzo sono divisi in quattro quadranti da 256k. Un registro di controllo per ciascun quadrante determina quali chip e quali uscite sono abilitate e la scrittura è abilitata quando nel quadrante è impostata la lettura o scrittura.



Unità interfaccia di memoria

Tutti i segmenti, ad eccezione di XPC, sono ridimensionabili e alcuni possono essere ridotti a misura zero e quindi non essere presenti nella mappa di memoria.

XPC è compreso tra 0E000h e 0FFFFh. Il segmento stack è compreso tra il valore indicato dai 4 bit superiori del registro SEGSIZE e 0DFFFh. Se i 4 bit sono settati a 0Eh o più, allora il segmento stack scompare.

I 4 bit inferiori del registro SEGSIZE determinano l' inizio del segmento dati. Anche in questo caso, se sono settati a 0Eh, il segmento scompare.

Un registro di sola scrittura permette di controllare le funzioni dei banchi di memoria.

Bits 7,6	Bit 5	Bit 4	Bit 3	Bit 2	Bits 1,0
00--4 stati d'attesa 01--2 stati d'attesa 10--1 stati d'attesa 11--0 stati d'attesa	1—Inversione A19	1—Inversione A18	1—Protezione da scrittura	0—sel. /OE0, /WE0 1—sel. /OE1, /WE1	00—sel. /CS0 01—sel. /CS1 1x—sel. /CS2

Registro di controllo dei banchi di memoria

I bit del registro hanno il seguente significato:

Bit 7,6: indicano gli stati di attesa;
 Bit 5,4: inversione delle linee A19 e A18 (permette di installare moduli da 1M nello spazio normalmente allocato ai 256k);
 Bit 3: protegge la memoria dalla scrittura;
 Bit 2: seleziona quale delle due linee OEX o WEX deve essere selezionata;
 Bit 1,0: determina quale chip deve essere selezionato.
 Durante il reset sono inizializzati tutti a zero.

Porte parallele

Il Rabbit possiede cinque porte parallele da 8 bit: A, B, C, D ed E. I pin usati da ciascuna porta sono condivisi con numerose altre funzioni.

Porta A: è condivisa con l' interfaccia dati della porta seriale;

Porta B: condivisa con le linee di controllo della porta slave e con il clock di I/O per le opzioni di temporizzazione del modo seriale per le porte seriali A e B;

Porta C: condivisa con la porta seriale;

Porta D: 4 bit sono condivisi alternativamente con i pin di I/O per le porte seriali A e B. 4 bit non sono condivisi. La porta D ha la possibilità di configurare le sue uscite come 'open drain' (test della matrice). La porta D ha in uscita dei registri di precaricamento che possono essere temporizzati nei registri d' uscita sotto il controllo del timer che genera la pulsazione. I bit 0-3 della porta D possono pilotare grosse correnti.

Porta E: tutti i bit della porta E possono essere configurati come strobe di I/O. 4 bit possono essere usati come input per gli interrupt esterni. Un bit è condiviso con il chip select della porta slave. Ha in uscita dei registri di precaricamento che possono essere temporizzati nei registri d' uscita sotto il controllo del timer che genera la pulsazione. Anche la porta E è in grado di pilotare grosse correnti.

Porta parallela A

E' formata da un unico registro di lettura/scrittura. Poiché viene usato dalla porta slave come porta dati, non deve essere usato se la porta slave è abilitata. Il registro di controllo della porta slave è usato per settare la porta parallela A come un input o un output. Impostando in esso il valore 080h, la porta si comporta come input, mentre impostando il valore 084h sarà un output.

Durante il reset viene impostata automaticamente come input.

Porta parallela B

E' formata da un registro con 6 bit di input e due di output solo se usata come porta parallela.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read	Echo drive	Echo drive	PB5 in	PB4 in	PB3 in	PB2 in	PB1 in	PB0 in
Write	PB7	PB6	x	x	x	x	x	x

Registro dati della porta parallela B (PBDR)

Se la porta slave è abilitata, i bit da PB2 a PB7 sono assegnati ad alcune funzioni per la porta slave. E' possibile leggere i valori di PB0 -> PB7 tramite il registro della porta dati anche quando PB2 -> PB7 sono usati dalla porta slave.

Durante il reset i pin PB6 e PB7 hanno valore zero.

Porta parallela C

E' formata da un registro con 4 pin di input e 4 di output. Quando vengono letti i dati del registro, i bit 1,3,5 e 7 restituiscono il valore della tensione applicata sui pin e i bit 0,2,4 e 6 restituiscono il valore del segnale che pilota i buffers di uscita. Questi due valori sono di norma gli stessi.

Questi pin sono pilotati dal registro dati della porta C o dalle linee di trasmissione di una delle porte seriali. Questa informazione è data dal settaggio dei bit nel registro PCFR.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCDR (r) adr = 050h	PC7 in	Echo drive	PC5 in	Echo drive	PC3 in	Echo drive	PC1 in	Echo drive
PCDR (w) adr = 050h	x	PC6	x	PC4	x	PC2	x	PC0
PCFR (w) adr = 055h	x	Drive TXA	x	Drive TXB	x	Drive TXC	x	Drive TXD

Registro dati (PCDR) e registro funzioni (PCFR) della porta parallela C

La porta parallela C condivide i pin con le quattro porte seriali e gli input possono servire come input per la porta seriale. Le porte seriali A e B possono usare alternativamente i pin 5 e 7 rispettivamente sulla porta D come input, e la sorgente dell' input della porta seriale per questa porta seriale dipende da come è settato il corrispondente registro di controllo delle porte seriali. Quando servono come input seriali, le linee dati possono essere lette dal registro dati della porta parallela C. Settando PCFR è possibile selezionare le uscite della porta parallela come uscite per le porte seriali, e in questo caso il valore del registro dati è ignorato. Durante il reset i bit attivi del registro funzioni e i bit del registro dati sono zero. Si avrà così zero sui quattro bit d' uscita.

Porta parallela D

Ha otto pin programmabili individualmente come ingresso o come uscita. Se programmati come uscite possono assumere il valore di uscite standard o uscite 'open drain'.

I registri d' uscita sono posti in cascata e controllati dal timer in modo da generare pulsazioni precise. Le uscite della porta D hanno un' elevata capacità di pilotaggio. I pin 4 e 5 possono essere usati come bit alternativi per la porta seriale B e i 6 e 7 per la porta seriale A..

Durante il reset il PDDDR è postato a zero ponendo tutti i pin come input. Inoltre i bit 0,1,4 e 5 del registro di controllo sono zero per assicurare che i dati siano temporizzati nei registri d' uscita quando vengono caricati. Gli altri registri associati alla porta D non vengono inizializzati durante il reset.

Bits 7, 6	Bits 5, 4	Bits 3, 2	Bits 1, 0
x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

Registro di controllo della porta parallela D

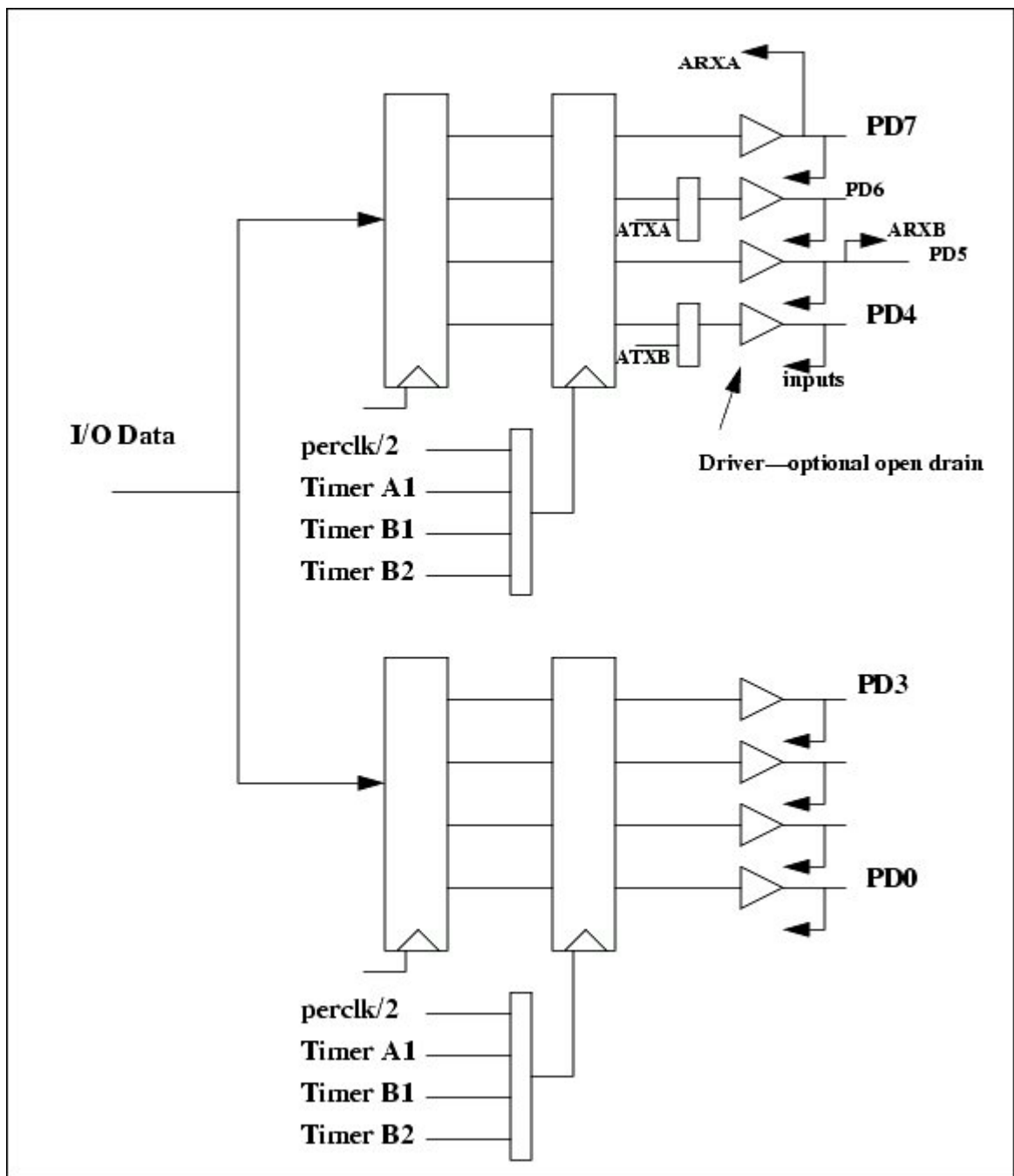


Diagramma a blocchi della porta parallela D

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PDDR (R/W) adr = 060h	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDDCR (W) adr = 066h	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain
PDFR (W) adr = 065h	x	alt TXA	x	alt TXB	x	x	x	x
PDDDR (W) adr = 067h	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out
PDB0R (W) adr = 068h	x	x	x	x	x	x	x	PD0
PDB1R (W) adr = 069h	x	x	x	x	x	x	PD1	x
PDB2R (W) adr = 06Ah	x	x	x	x	x	PD2	x	x
PDB3R (W) adr = 06Bh	x	x	x	x	PD3	x	x	x
PDB4R (W) adr = 06Ch	x	x	x	PD4	x	x	x	x
PDB5R (W) adr = 06Dh	x	x	PD5	x	x	x	x	x
PDB6R (W) adr = 06Eh	x	PD6	x	x	x	x	x	x
PDB7R (W) adr = 06Fh	PD7	x	x	x	x	x	x	x

Registri della porta parallela D

Porta parallela E

E' formata da 8 pin programmabili individualmente come input o come output. PE7 è utilizzato come chip select della porta slave quando questa è abilitata. Ciascuna uscita può essere configurata come strobe di I/O. Inoltre quattro delle linee della porta E possono essere usate come input per le richieste di interrupt.

I registri d' uscita sono posti in cascata e controllati da timer.

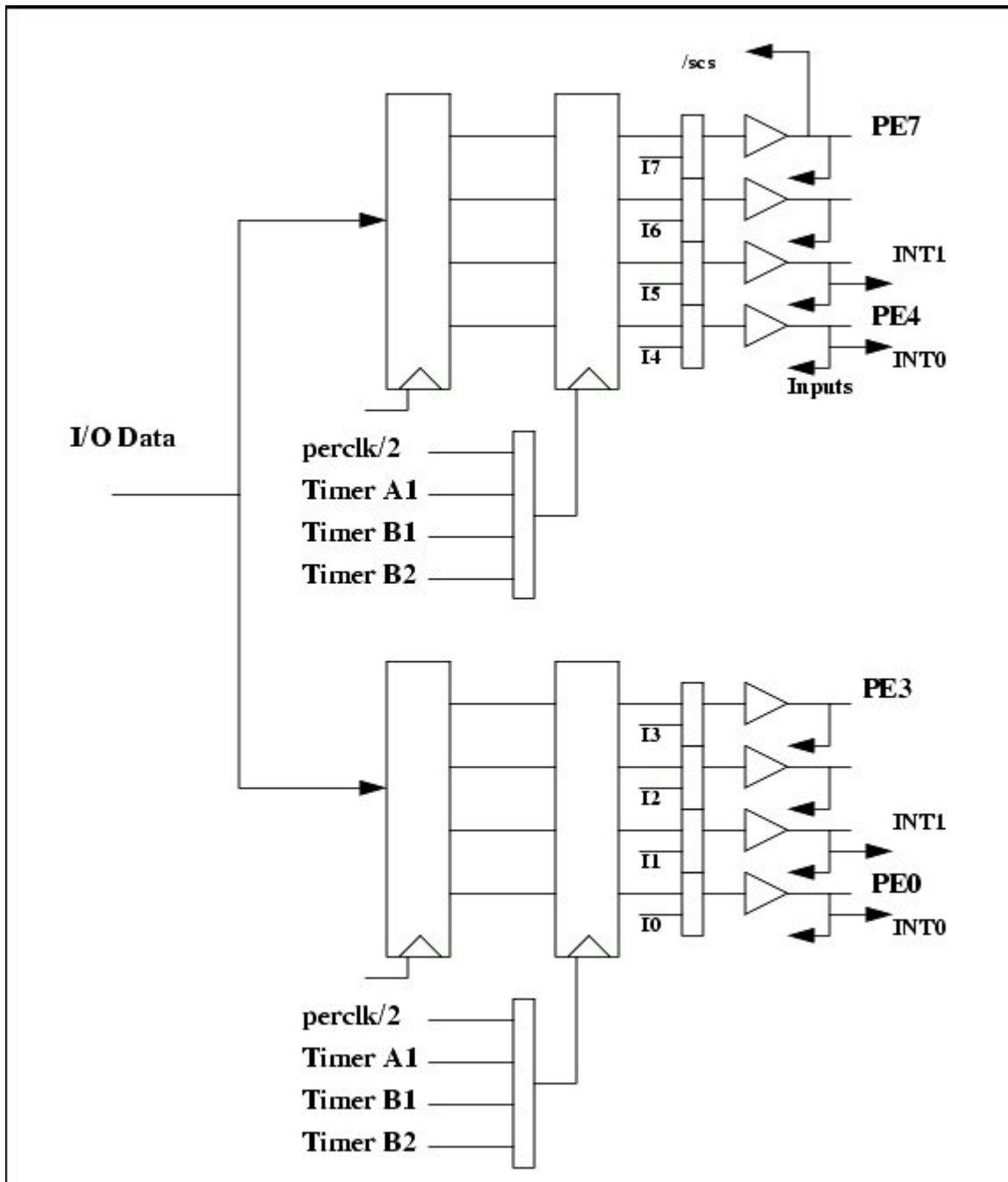


Diagramma a blocchi della porta E

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDR (R/W) adr = 070h	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEFR (W) adr = 075h	alt /I7	alt /I6	alt /I5	alt /I4	alt /I3	alt /I2	alt /I1	alt /I0
PEDDR (W) adr = 077h	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out
PEB0R (W) adr = 078h	x	x	x	x	x	x	x	PE0
PEB1R (W) adr = 079h	x	x	x	x	x	x	PE1	x
PEB2R (W) adr = 07Ah	x	x	x	x	x	PE2	x	x
PEB3R (W) adr = 07Bh	x	x	x	x	PE3	x	x	x
PEB4R (W) adr = 07Ch	x	x	x	PE4	x	x	x	x
PEB5R (W) adr = 07Dh	x	x	PE5	x	x	x	x	x
PEB6R (W) adr = 07Eh	x	PE6	x	x	x	x	x	x
PEB7R (W) adr = 07Fh	PE7	x	x	x	x	x	x	x

Registri della porta parallela E

Bits 7, 6	Bits 5, 4	Bits 3, 2	Bits 1, 0
x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

Registro di controllo della porta parallela E

Porte seriali

Il μP è fornito di quattro porte seriali: A, B, C e D che possono gestire comunicazioni seriali asincrone ad elevato trasferimento.

Le porte A e B possono inoltre operare come porte temporizzate e come porte alternative di I/O.

Ogni porta seriale può operare a 500kbps in modalità asincrona e a 62kbps in modalità sincrona.

Ciascuno dei 7 o 8 bit di dati viene trasmesso in modalità asincrona. E' supportata la modalità 9^{no} bit e bit di indirizzo. I bit di stop e di parità non sono supportati dall' hardware, ma possono essere gestiti in modo software

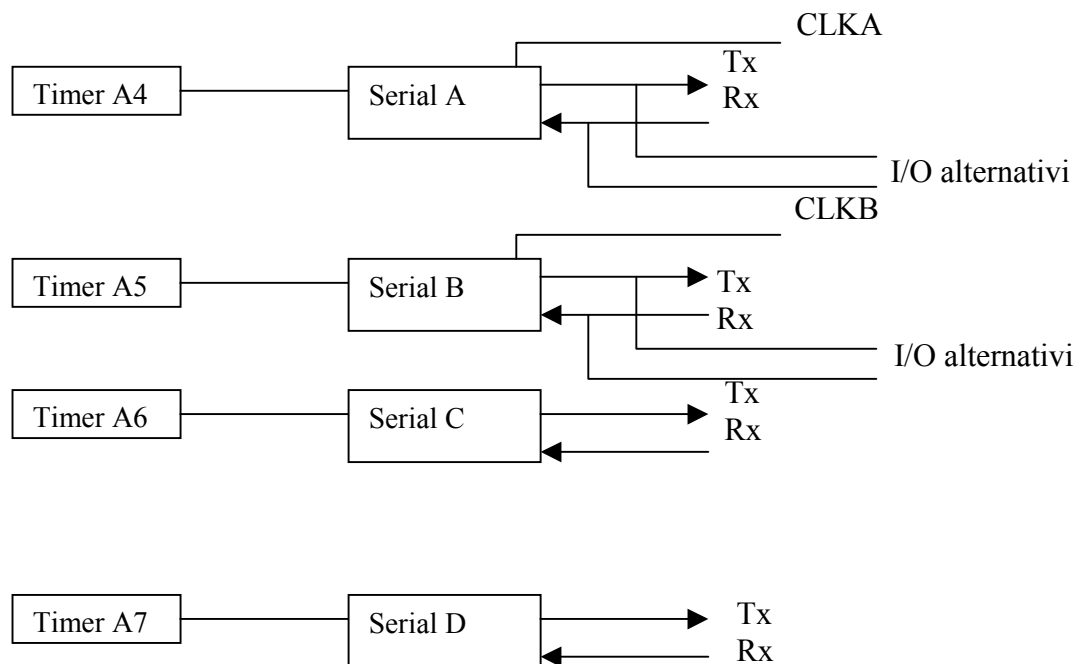


Diagramma a blocchi delle porte seriali

Registri delle porte seriali

La precedente figura mostra il diagramma a blocchi della porta seriale. Ciascuna porta ha un registro dati, un registro di controllo e un registro di stato. La scrittura nel registro dati avvia la trasmissione. Se i dati vengono ricevuti, questi vengono inseriti nel registro dati.

Il registro di controllo è usato per settare i parametri di trasmissione e ricezione.

Il clock di ingresso può essere 16 volte maggiore del baud rate in modalità asincrona e 2 volte maggiore in modalità temporizzata quando viene usato il clock interno.

I timers A4 – A7 gestiscono il clock di ingresso per le porte, e possono dividerne la frequenza di un valore compreso tra 1 e 256.

Register	Address xx = 00, 01, 10, 11 for A, B, C, D	Mnemonic x = A, B, C, D
Registro dati	11xx0000	SxDR
Registro dati alternativo per l' invio del 9 ^{no} bit d' indirizzo	11xx0001	SxAR
Registro di stato	11xx0011	SxSR
Registro di controllo (sola scrittura)	11xx0100	SxCR

Registri delle porte seriali

Registro di stato

Bit 7: ricevitore pronto – viene settato quando un byte è trasferito dallo shift register al registro dati del ricevitore. Quando quest' ultimo viene letto, il bit 7 è settato a zero. La transizione da 0 a 1 setta il flip flop del ricevitore per la richiesta di interrupt.

Bit 6: bit di indirizzo o 9^{no} bit – viene settato se il carattere nel registro dati del ricevitore ha il 9^{no}. Viene posto a zero prima della lettura del registro dati finchè un nuovo dato con un nuovo bit di indirizzo viene caricato.

Bit 5: è settato se il ricevitore è sovraccarico. Viene posto in attesa se lo shift register e il registro dati sono pieni e viene rilevato un bit di start. Quando il registro dati è letto, questo bit è posto a zero.

Bit 3: - buffer dati del trasmettitore pieno – è settato quando il registro dati del trasmettitore è pieno e quindi un byte è scritto nel registro dati della porta seriale. E' posto a zero quando un byte è trasmesso allo shift register del trasmettitore o un' operazione di scrittura è eseguita nel registro di stato. Richiede un' interrupt nella transizione da 0 a 1.

Bit 2: - bit di occupato del trasmettitore – viene settato se lo shift register del trasmettitore è occupato a trasmettere un dato. E' settato nella fase di discesa del bit di start, che è inoltre il clock che trasferisce i dati dal registro dati del trasmettitore allo shift register. Viene posto a zero alla fine di ogni bit di stop.

Bit 0, 1, 4: sono posti sempre a zero.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1,0
Ricevitore pronto	9 ^{no} bit ricevuto	Biffer di ricezione sovraccarico	0	Registro dati di trasmissione pieno	Il trasmettitore stà inviando un byte	0,0

Registro di stato delle porte seriali

Registro di controllo

Bit 7, 6: in modalità asincrona sono sempre zero. Per le porte A e B, se è abilitata la modalità sincrona, bisogna settare il codice riportato nella successiva tabella.

Bit 5, 4: abilita i pin.

Bit 3, 2: setta la modalità operativa.

Bit 1, 0: abilita gli interrupt e setta le priorità.

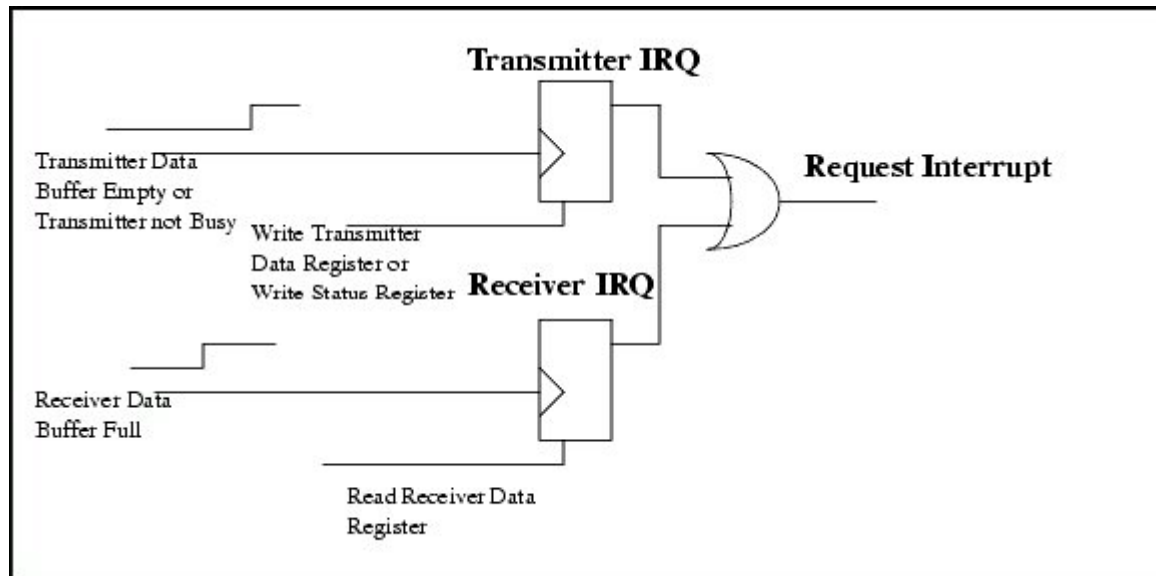
Bit 7,6	Bit 5,4	Bit 3,2	Bit 1,0
00--no op 01--receive 1 byte clocked mode (A,B) 10--send one byte clocked mode (A,B) 11--reserved for future use	00--use port C for serial input 01--use port D for serial input 1x--disable receiver input	00--async mode, 8 bits 01--async mode 7 bits 10--clocked mode external clock (A,B) 11--clocked mode internal clock (A,B)	00--no interrupt 01-- priority 1 interrupt 10--priority 2 11--priority 3

Registro di controllo della porta seriale

Interrupt della porta seriale

Per ricevere e inviare un interrupt si usa lo stesso vettore. I flip flop di interrupt sono due, uno per trasmettere e l' altro per ricevere. Se entrambi sono settati, è richiesto un interrupt. Vengono settati solo durante la fase di salita e posti a zero da una pulsazione generata da un' operazione di I/O in lettura o scrittura. Quando è richiesto un interrupt, esso viene acquisito subito dopo che viene completata l' operazione in corso e seguendo le priorità. L' interrupt decade se la richiesta è fatta

dopo il campionamento del bit di stop in ricezione, generalmente a metà del bit di stop. I bit di dati sono trasmessi su questo stesso clock dallo shift register del ricevitore al registro dati del ricevitore. Il flip flop del trasmettitore che richiede l'interrupt viene settato in fase di discesa del bit di stop quando il registro dati è vuoto e in fase di salita se è vuoto lo shift register. Finché il registro dati è vuoto durante la fase di salita, il trasmettitore non è pronto. Lo è solo se il registro dati è vuoto durante la fase di salita del bit di stop.



Generazione degli interrupt delle porte seriali

Tempi di trasmissione dei dati seriali

Durante la trasmissione, se gli interrupt sono abilitati, un interrupt viene richiesto quando il registro di trasmissione si svuota e, inoltre, un interrupt si verifica quando shift register e registro di trasmissione si svuotano entrambi.

Quando il registro dati di trasmissione contiene dati e lo shift register ha ultimato l'invio, i bit di dati vengono temporizzati dal registro di trasmissione allo shift register, e lo shift register è nuovamente pronto. La richiesta di interrupt è disabilitata dalla scrittura sul registro dati o sul registro di stato. Durante la routine di servizio di un interrupt, tutti i dati vengono salvati nel registro dati. Dopo che la routine è stata completata, il sistema viene riportato nello stato precedente alla richiesta.

Tempi di ricezione dei dati seriali

Quando il ricevitore è pronto, una linea di discesa indica l'inizio della trasmissione. Quando viene rilevato il bit di start, i bit di dati sono campionati a metà di ciascun bit di dati e vengono spostati nello shift register di ricezione. Dopo 7 o 8 bit di dati, il successivo (9^{no}) è il bit di indirizzo o il bit di stop. Se la linea Rx è bassa, esso è un bit di indirizzo e il ricevitore deve attendere il campionamento del bit di stop. Se la linea è alta durante il campionamento, esso è un bit di stop. I bit di dati vengono trasferiti nel registro dati del ricevitore e un interrupt, se abilitato, viene richiesto.

In ricezione, viene richiesto un interrupt quando il registro dati del ricevitore ha la disponibilità dei dati. La trasmissione dallo shift register al registro dati setta il bit 7 del registro di stato. La richiesta di interrupt e il bit 7 vengono resettati quando il registro dati viene letto.

Se il bit 7 è alto, viene richiesto un interrupt.

Porte seriali temporizzate

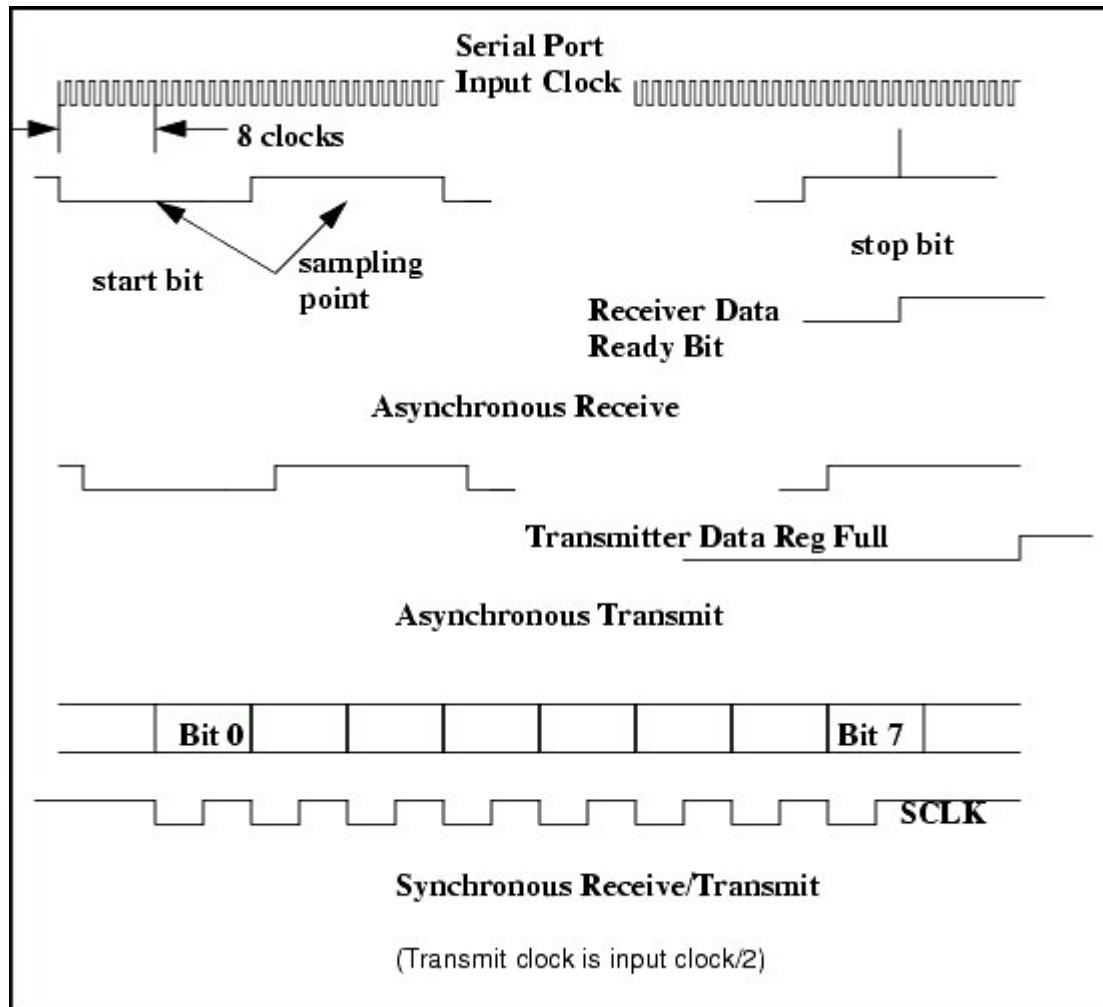
Le porte A e B possono operare in modalità temporizzata. I dati e il clock vengono forniti da una serie di 8 bit. Lo shift register di trasmissione avanza sulla linea di discesa del clock. Il ricevitore campiona i dati sulla linea di salita del clock. La porta seriale può generare il clock o questo può essere fornito dall'esterno.

Per abilitare la temporizzazione bisogna settare i bit 3 e 2 del registro di controllo. Lo shift register e il registro dati lavorano nello stesso modo come per le comunicazioni asincrone. Per inizializzare l'invio o la ricezione, il codice deve essere settato (bit 7 e 6) nel registro di controllo per ciascun byte.

Per trasmettere utilizzando il clock interno, bisogna prima caricare il registro dati e poi salvare il codice inviato. Quando è completato l'invio, il registro dati deve essere caricato nello shift register e trasmesso.

Per ricevere, lo shift register del ricevitore deve essere pronto. Bisogna salvare il codice ricevuto nel registro di controllo. Viene generata una serie di otto cicli di clock e chi invia deve rilevare il clock ed espellere i dati sulla linea dati durante la fase di discesa di ciascun clock.

Il ricevitore campiona i dati sulla linea di salita.



Sincronizzazione delle porte seriali

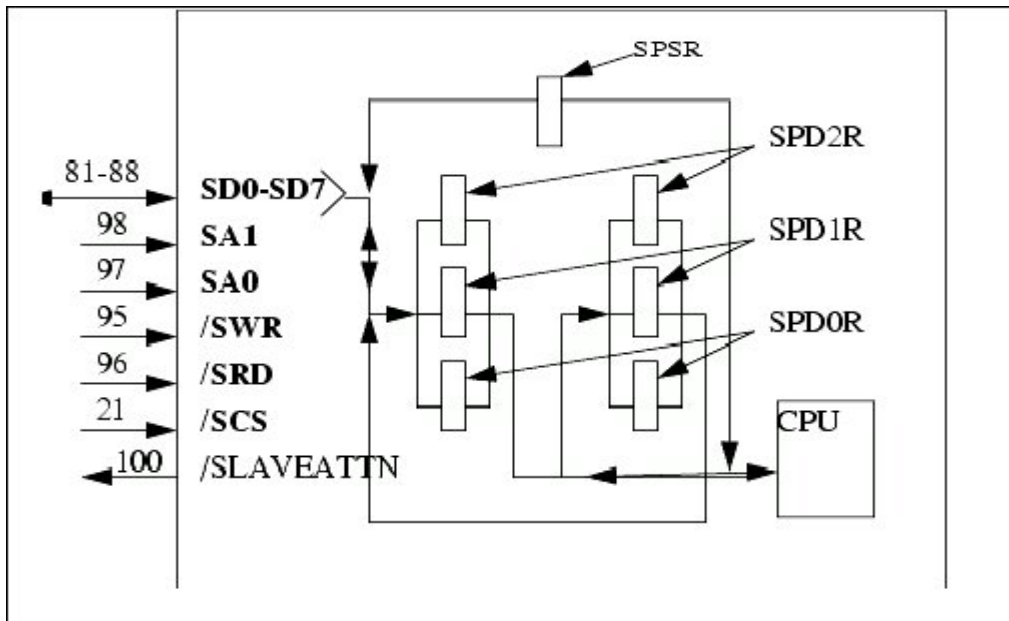
Tempi delle porte seriali

Per le comunicazioni sincrone, il clock può essere generato dal Rabbit o da una periferica esterna. In quest'ultimo caso il segnale di clock deve essere sincronizzato con la periferica interna che genera il clock prima di trasmettere o ricevere.

La porta seriale può essere usata per generare un interrupt. Inoltre alcuni sistemi richiedono due bit di stop, il bit di parità o il 9^{no} bit di dati per segnare l'inizio del frame.

Porta slave

Quando il μP è configurato come slave, la porta parallela A e altre linee dati sono usate per comunicare dati tra master e slave. La porta slave è un dispositivo costituito da registri dati, un bus dati e altre linee di controllo.



Porta slave

Essa è costituita da tre registri dati per ciascuna direzione di comunicazione. I registri SPD0R, SPD1R ed SPD2R possono essere scritti dal master e letti dallo slave. La stessa cosa succede per gli altri registri nell'altra direzione di comunicazione. Lo stato del registro può essere letto sia dal master che dallo slave. Esso può essere pieno o vuoto. Il registro dati è considerato pieno quando è scritto da entrambi i lati. Se il registro può essere letto da entrambi i lati allora è considerato vuoto. Il flag è settato a 1 quando viene scritto e a zero se viene letto. Essi appaiono come registri interni di I/O per lo slave ed esterni per il master.

Timers

IL Rabbit è provvisto di due timers: A e B. A è usato per generare il baud clock per le porte seriali, un clock periodico per temporizzare le porte parallele D ed E o per generare gli interrupt periodici. Il timer B può gestire le stesse funzioni di A tranne che generare il baud clock.

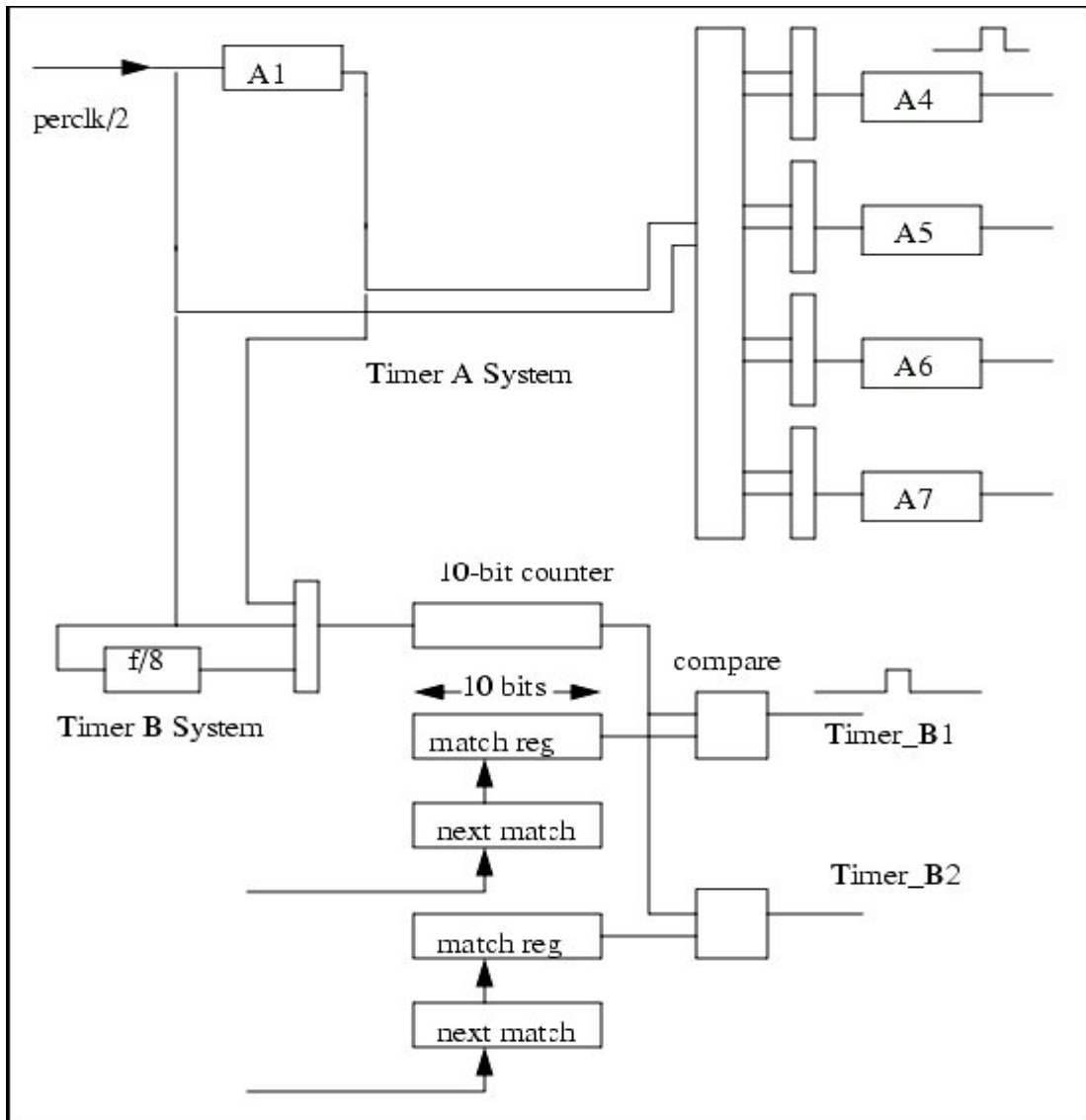


Diagramma a blocchi dei timer

Timer A

E' formato da cinque timers separati: A1, A4, A5, A6 ed A7 che sono tutti registri a 8 bit

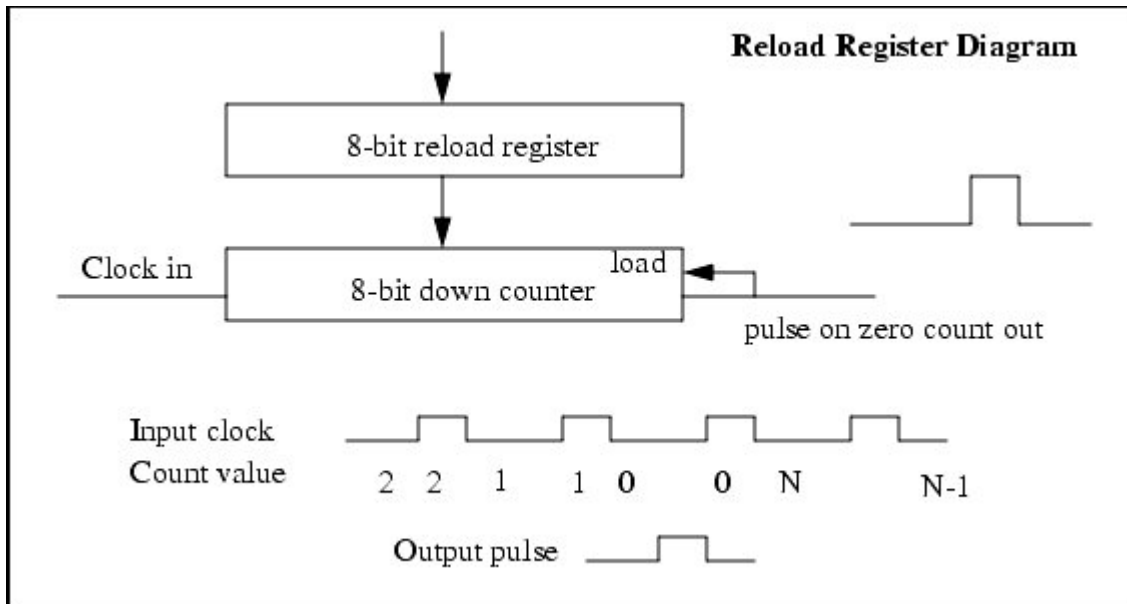


Diagramma a blocchi del registro di reload

Il registro di reload può contenere un numero tra 0 e 255. Il contatore divide questo valore per $(n+1)$. Per esempio se esso contiene il valore 127, allora dalla sinistra del contatore entrano 128 pulsazioni che escono da destra. Se il registro di reload contiene zero, allora per ciascuna pulsazione in entrata ce n'è una in uscita.

I timer sono pilotati dal clock di periferica diviso per due. Esso è lo stesso clock del processore, o questo diviso per 8. Le pulsazioni d'uscita sono lunghe un clock. La temporizzazione del contatore avviene durante la fase di discesa negativa di questa pulsazione.

Quando il contatore arriva a zero, il reload register è caricato sulla successiva pulsazione d'ingresso. Inoltre esso viene ricaricato a ogni istante fino a che la periferica del clock è sincrona con il clock del processore.

I timer A4, A5, A6 e A7 generano il baud clock delle porte seriali A, B, C e D rispettivamente. Ciascuno dei 5 registri di countdown può generare un interrupt. Un registro di stato comune ha un bit per ciascun timer che indica se la pulsazione d'uscita per questi timer è stata presa dopo l'ultima lettura del registro di stato. Quando il registro di stato viene letto, questi bit sono resettati. Essi possono essere letti da una lettura sul registro di stato e settati dopo il completamento di tale lettura. Se un bit è acceso e il corrispondente interrupt è abilitato, viene richiesto un interrupt.

Registri di I/O del timer A

Register Name	Register Mnemonic	I/O address (hex)	R/W
Timer A Control/Status Register	TACSR	A0	R/W
Timer A Control Register	TACR	A4	W
Timer A1 Time Constant 1 Register	TAT1R	A3	W
Timer A4 Time Constant 4 Register	TAT4R	A9	W
Timer A5 Time Constant 5 Register	TAT5R	AB	W
Timer A6 Time Constant 6 Register	TAT6R	AD	W
Timer A7 Time Constant 7 Register	TAT7R	AF	W

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Read	A7 count done	A6 count done	A5 count done	A4 count done	0	0	A1 count done	This bit is write only.
Write	A7 interrupt enable	A6 interrupt enable	A5 interrupt enable	A4 interrupt enable	x	x	A1 interrupt enable	1--enable Timer A

Registri di controllo e di stato del timer A

Bit 7 A7	Bit 6 A6	Bit 5 A5	Bit 4 A4	Bits 3, 2	Bits 1, 0
Source A7 0-pclk/2 1-A1	Source A6 0-pclk/2 1-A1	Source A5 0-pclk/2 1-A1	Source A4 0-pclk/2 1-A1	not used ignored	00--Interrupt disabled 01--Enable priority 1 interrupt 10--Enable priority 2 interrupt 11--Enable priority 3 interrupt

Registro di controllo

Bit 1, 4, 7: lettura/scrittura. Leggendo questo registro si resetta ogni bit (1, 4, 5, 6, e 7). Scrivendo si abilitano gli interrupt per i corrispondenti timer.

Bit 0: scrittura. Se settato a 1 abilita il clock (PERCLK/2) per il timer A, se a 0 lo disabilita.

Il timer A è disabilitato durante l' accensione. In generale è settato mentre il clock è disabilitato, ma può cambiare settaggio anche mentre il clock è in funzione.

Timer B

Il contatore del timer B può essere pilotato direttamente da PERCLK/2, da (PERCLK/2)/8 oppure dall' uscita del timer A1. Il timer B ha un contatore a 10 bit in continua esecuzione. Esso è comparato con due match register, B1 e B2.

Quando il contatore ha un valore uguale a quello del match register , viene generata una pulsazione interna con una lunghezza di un clock. La pulsazione del match register può essere usata per generare un interrupt e/o un clock sui registri d' uscita delle porte parallele D ed E.

I match register sono caricati dai preload register che vengono scritti da un' istruzione di I/O. I byte di dati nel preload register avanzano nel successivo match register quando viene generata una pulsazione dal match register. Ogni volta che avviene una condizione di match, il processore setta un bit interno che segna un valore nel TBLxR come non valido. Leggendo TBCSR di resetta la condizione di interrupt.

TBLxR deve essere ricaricato per riabilitare l' interrupt.

TBMxR non deve essere ricaricato ogni volta.

Register Name	Register Mnemonic	I/O Address (hex)	R/W	On Reset To
---------------	-------------------	-------------------	-----	-------------

Timer B Control/Status Register	TBCSR	B0	R/W	xxxxxx000
Timer B Control Register	TBCR	B1	W	xxxxxx00
Timer B MSB 1 Reg	TBM1R	B2		x
Timer B LSB 1 Reg	TBL1R	B3	W	x
Timer B MSB 2 Reg	TBM2R	B4	W	x
Timer B LSB 2 Reg	TBL2R	B5	W	x
Timer B Count MSB Reg	TBCMR	BE	R	x
Timer B Count LSB Reg	TBCLR	BF	R	x

Registri del timer B

Bits 7:3	Bit 2	Bit 1	Bit 0
Not used	1--A match with match register 2 was detected. This bit is cleared when this register is read; setting this bit to 1 enables the interrupt.	1--A match with match register 1 was detected. This bit is cleared when this register is read; setting this bit to 1 enables the interrupt.	1--Enable the main clock for this timer.

Registri di stato e di controllo

Bits 7:4	Bits 3:2	Bits 1:0
Not used	00--Counter clocked by perclk/2 01--Counter clocked by output of timer A1 1x--Timer clocked by perclk/2 divided by 8	00--Interrupt disabled xx--Interrupt priority xx enabled.

Registro di controllo

Normalmente viene generato un interrupt quando entrambi i comparatori del timer B generano una pulsazione. La routine di interrupt deve vedere quale dei due comparatori ha richiesto l' interrupt. La routine setta il valore successivo. Se sono utilizzate le porte parallele temporizzate, i valori vengono caricati in alcuni bit del registro della porta parallela.

Appendice

Descrizione del bios

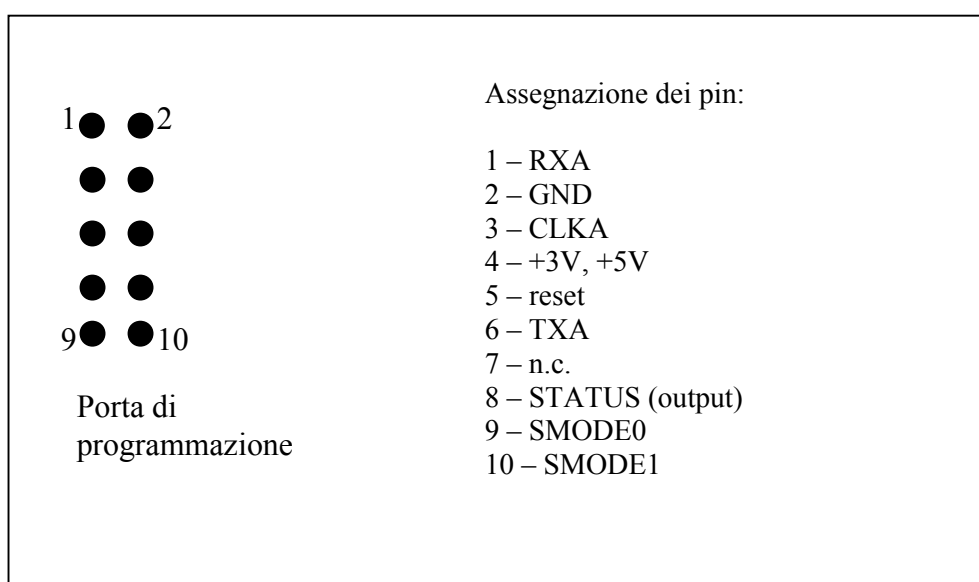
Il bios viene compilato prima di ogni altro programma. Esso occupa lo spazio all' inizio del segmento root. Se un programma utente inizia all' indirizzo zero, allora esso inizia nel bios. Non ci sono limiti sulla quantità di codice da includere nel bios.

Il bios supporta i seguenti servizi:

- avvio del sistema, che include setup della memoria, stati di attesa e velocità di clock;
- lettura e programmazione del Real Time Clock;
- operazione e gestione degli interrupt periodici;
- gestione del consumo;
- gestione del modo operativo delle porte parallele e ingresso/uscita dei dati da esse;
- routine per l' inizializzazione e manipolazione dei timers;
- routine per la scrittura della memoria flash;
- servizi base per il multitasking;
- routine per il supporto dei parametri di gestione della porta di programmazione;

Porta di programmazione

Gestisce il dialogo diretto tra la board e la piattaforma di programmazione Dynamic C.



Il pin STATUS è utilizzato per richiedere attenzione quando viene incontrato un breakpoint durante il test.

I pin SMODE0 ed SMODE1 sono posti a tensione di +5V/+3V. La board fornisce i +3V/+5V necessari per pilotare l' interfaccia RS232 sul PC. RXA e TXA sono i pin di ricezione e trasmissione.

CLKA è il clock.

Uso come porta diagnostica/ di setup

La porta di programmazione può servire come porta di comunicazione per i test e la diagnosi del sistema. Una volta connessa la board al PC tramite la seriale, il pin Reset può essere settato manipolando l' interfaccia RS232 sul PC.

Il PC può riavviare il bios tramite l' invio della stringa 24h 80h 80h. Subito dopo viene verificato, poiché i pin SMODE0 ed SMODE1 sono ad 1, il collegamento del cavo di programmazione. Ciò provoca l' ingresso in modalità programmazione.

La porta di programmazione utilizza la porta seriale A. E' possibile usare come porta alternativa la porta parallela e l'uso dei pin 6 e 7 della porta parallela C come TXA ed RXA. Ciò permette di usare la porta di comunicazione in modalità sincrona.

Set di istruzioni

Istruzioni per il caricamento immediato di dati in un registro

La costante che segue il codice operativo nel flusso di istruzioni può essere generalmente caricata in un registro tranne che in PC, AF, IP ed F (il caricamento di PC è un'istruzione di salto).

Alcuni esempi sono:

LD A, 3	carica il valore 3 nel registro A (8 bit)
LD HL, 256	carica il valore 256 nel registro HL (16 bit)
LD BC', 3567	
LD IX, 1234	

Istruzioni per il caricamento o memorizzazione dei dati da o in un indirizzo costante

LD A, (mn)	carica 8 bit dall'indirizzo mn
LD (mn), A	carica l'indirizzo mn con 8 bit
LD HL, (mn)	carica 16 bit dall'indirizzo mn
LD (mn), A	carica l'indirizzo mn con 16 bit

Allo stesso modo si possono caricare valori a 16 bit in e da registri quali DE, BC, SP, IX e IY. E' possibile anche caricare dati nei registri alternativi, ma non è possibile salvare dati dai registri alternativi direttamente in memoria.

LD A', (mn)	carica 8 bit dall'indirizzo mn
LD (mn), A'	istruzione non ammessa

Istruzioni per il caricamento o il salvataggio di dati con l'utilizzo di un registro indice

Un registro indice è un registro a 16 bit, generalmente IX, IY, SP o HL che è usato per l'indirizzamento di un byte o di una parola prelevati o caricati dalla memoria.

BC e DE possono essere usati come registri indice solo per i casi seguenti:

LD A, (BC)	
LD A', (BC)	
LD HL, (mn)	
LD (BC), A	
LD A, (DE)	
LD A', (DE)	
LD (DE), A	

Le istruzioni sono le seguenti:

LD r, (HL)	r = A; B; C; D; E; H; L;
LD r', (HL)	r' è un registro alternativo
LD (HL), r	
LD (HL), r'	nom è ammessa

LD r, (HL)	r = A; B; C; D; E; H; L;
LD r', (HL)	r' è un registro alternativo
LD (HL), r	
LD (HL), r'	non è ammessa
LD r, (IX+d)	r (o r') è uno dei registri visti, d è un offset compreso tra -128 e 127
LD (IX+d), r	
LD (IY+d), r	IX o IY hanno d come offset
LD HL, (SP+d)	d compreso tra 0 e 255
LD (SP+d), HL	

Istruzioni di movimento tra registri

E' possibile muovere dati tra registri a 8 bit A, B, C, D, E, H, L:

LD A, C	sposta il valore di C in A
LD D, E	

I registri alternativi possono essere registri di destinazione:

LD A', C
LD D', E

Le istruzioni appena viste richiedono 2 bytes e 4 cicli di clock

Non sono ammesse le istruzioni seguenti:

LD A, D'
LD D', E'

Inoltre è possibile muovere istruzioni a 16 bit:

LD dd', BC	dd' = HL', DE', BC'
LD dd', DE	
LD IX, HL	
LD SP, HL	
LD SP, IX	
LD SP, IY	

Scambio tra registri

Permette due movimenti con una sola istruzione:

EX af, af'	scambia il contenuto di af con af'
EXX	scambia HL, DE, BC con HL', DE', BC'
EX DE', HL	
EX DE', HL'	
EX DE', HL'	

Istruzioni di PUSH e POP

Le istruzioni PUSH e POP sono valide solo per AF, HL, DC, BC, IX e IY. Per i registri AF', HL', DE' e BC' è valida solo l'istruzione POP.

POP HL
PUSH BC
PUSH IX

Operazioni aritmetico logiche a 16 bit

Il registro HL è l' accumulatore principale a 16 bit. IX e IY possono servire come accumulatori alternativi per molte operazioni a 16 bit.

ADD HL, WW somma il contenuto di WW= HL, DE, BC, SP con HL
ADC HL, WW
INC WW incrementa WW
(HL può essere sostituito da IX o IY)

SHIFT:

RR HL ruota la destra di HL con il carry
RL DE ruota la sinistra di DE con il carry

Operazioni logiche:

AND HL, DE
AND IX, DE
OR HL, DE

BOOL HL testa il registro HL

Istruzioni di I/O

Ogni istruzione di accesso alla memoria è preceduta da due prefissi, uno per lo spazio interno di I/O e l' altro per quello esterno.

IOI LD A, (85h) carica il registro A con il contenuto del registro interno di I/O
posto alla locazione 85h
LD IY, 4000h carica IY con il valore 4000h
IOE LD HL, (IY+5) carica il registro HL con il contenuto del registro esterno di I/O
Alla locazione 4005h

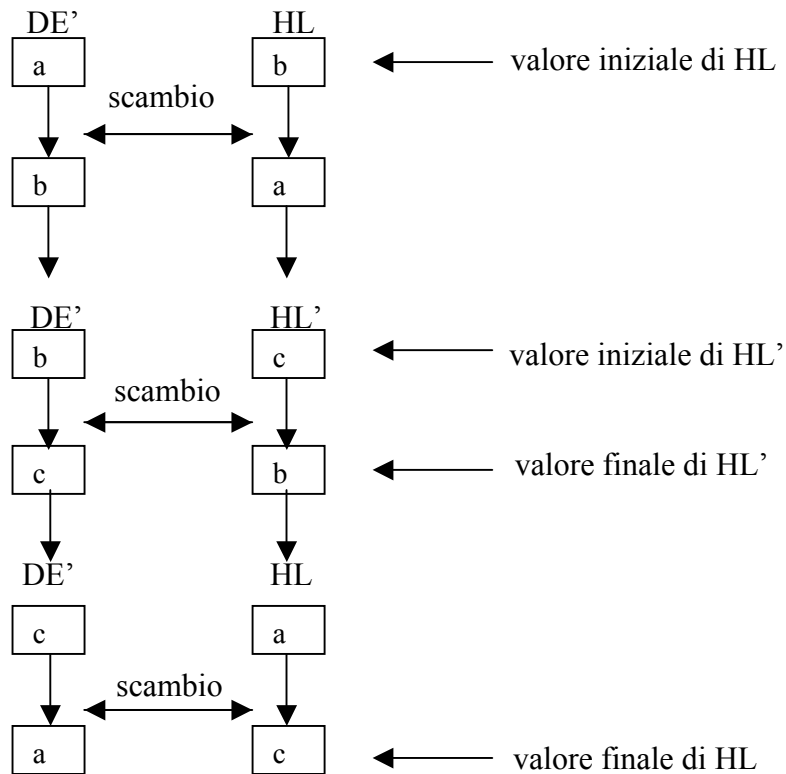
Le operazioni di scrittura di I/O dei registri interni richiede due clock, mentre la scrittura della memoria o periferiche esterne di I/O richiede almeno tre clock.

Scambi non direttamente implementati

Lo scambio tra alcuni registri non può essere effettuato direttamente. Si utilizza quindi uno scambio indiretto:

$HL \longleftrightarrow HL' \quad (8 \text{ clock})$
 $EX \ DE', HL \quad (2 \text{ clock})$
 $EX \ DE', HL' \quad (4 \text{ clock})$
 $EX \ DE', HL \quad (2 \text{ clock})$

Questo scambio può essere rappresentato meglio con uno schema a blocchi che mostra come effettivamente esso avviene:



Descrizione dei pin

Gruppo	Nome del pin	Direzione	Funzione	Numero del pin
Hardware	CLK	Output	E' il clock d' uscita. Viene preso dall' oscillatore principale di sistema (PERCLK) e può essere diviso per 8, raddoppiato o entrambi tramite un circuito programmabile interno. Viene abilitato dopo il reset. Sotto la guida di un software può portare in uscita la frequenza di clock piena, o dividerla per due o può essere utilizzato come pin d' uscita..	1
	/RESET	Input	Reset generale di sistema.	37
	XTALA1	Input	Quarzo per clock da 32 Mhz. Se si usa un clock esterno, il pin deve essere pilotato da questo clock.	40
	XTALA2	Output	Non connesso in caso di clock esterno.	41
	XTALB1	Input	Quarzo per clock principale. Se si usa un clock esterno, il pin deve essere pilotato da questo clo.	90
	XTALB2	Output	Non connesso in caso di clock esterno.	91
CPU Buses	A0-A19	Output	Bus indirizzi.	7, 17-20, 61-68, 70-75, 79
	D0-D7	Bidirectional	Bus dati.	9--16
Status/Control	/WDTOUT	Output	Quando il sistema watchdog v� fuori tempo, questo pin genera una pulsazione. Pu� essere usato per generare una pulsazione da 30 �s.	34
Status	STATUS	Output	Programmabile per diverse funzioni.	38
Status	SMODE1 SMODE0	Input	Selezione della modalit� di avvio. (SMODE1=0, SMODE0=0) l' esecuzione inizia all' indirizzo zero. (0;1) boot dalla porta slave (1;0) boot dalla porta seriale temporizzata A (1;1) boot dalla porta seriale asincrona A a 2400 bps Possono essere usati come input generici dopo il completamento del boot.	35-36 (1:0)
Chip Selects	/CS0	Output	Seleziona il chip di memoria 0. E' usato generalmente per selezionare la memoria in cui caricare il programma.	8
	/CS1	Output	Seleziona il chip di memoria 1.	5
	/CS2	Output	Seleziona il chip di memoria 2.	4
Output Enables	/OE0	Output	Abilita l' uscita della memoria flash.	6
	/OE1	Output	Abilita l' uscita della memoria SRAM.	76
Write Enables	/WE0	Output	Abilita la scrittura della memoria flash.	69
	/WE1	Output	Abilita la scrittura della memoria SRAM.	80
I/O Control	/BUFEN	Output	Abilita il buffer di I/O. E' pilotato basso durante un ciclo di I/O esterno (n.c.).	33
I/O Read Strobe	/IORD	Output	Strobe in lettura I/O. E' pilotato basso durante un ciclo esterno di lettura del bus (n.c.).	32
I/O Write	/IOWR	Output	Strobe in scrittura I/O. E' pilotato basso durante un ciclo	31

Strobe			esterno di scrittura del bus (n.c.).	
I/O Port A	PA0-PA7	Input/ Output	Porta parallela A. Funzioni generali di I/O oppure porta dati per la porta slave.	81-88
I/O Port B	PB0-PB7	6 In/2 Out	Porta parallela B. Se usata come porta parallela, PB7 e PB6 sono solo uscite e PB0, PB5 solo ingressi. PB0 e PB1 possono essere uscite quando settate come clock per le porte seriali. In reset le uscite sono settate a zero. Se è abilitata la porta slave hanno altre funzioni. Se la porta seriale A è abilitata in modo temporizzato, allora PB1 è la linea di clock bidirezionale. Se la porta seriale B è abilitata in modo temporizzato, allora PB0 è la linea di clock bidirezionale.	93-100
I/O Port C	PC0-PC7	4 In/4 Out	Porta parallela C. Se usata come porta parallela, i bit 1, 3, 5 e 7 sono input e i pin 0, 2, 4 e 6 sono output. I bit 0, 2, 4 e 6 possono essere usati come uscita seriale dei dati per le porte seriali D, C, B e A rispettivamente e i bit 1, 3, 5 e 7 come input.	51, 54-60
I/O Port D	PD0-PD7	Input/ Output/ output open drain	Porta parallela D. Ciascun bit può essere selezionato come input o come output. Ciascuna uscita può essere selezionata come open drain. PD6 può essere programmato come un'uscita seriale opzionale per la porta seriale A e PD 4 per la porta B. PD7 e PD5 possono essere usati come ingressi seriali per le porte seriali A e B.	43-50
I/O Port E	PE7-PE0	Input/ Output	Porta parallela E. Ciascun bit può essere selezionato come input o output.	21-26, 29, 30
Power	VBAT		+3.0 V (batteria di backup), +3.3 V o +5.0 V	42
	VDD		+3.3 V o +5.0 V	3, 28, 53, 78, 92
	VSS		Massa	2, 27, 39, 52, 77, 89