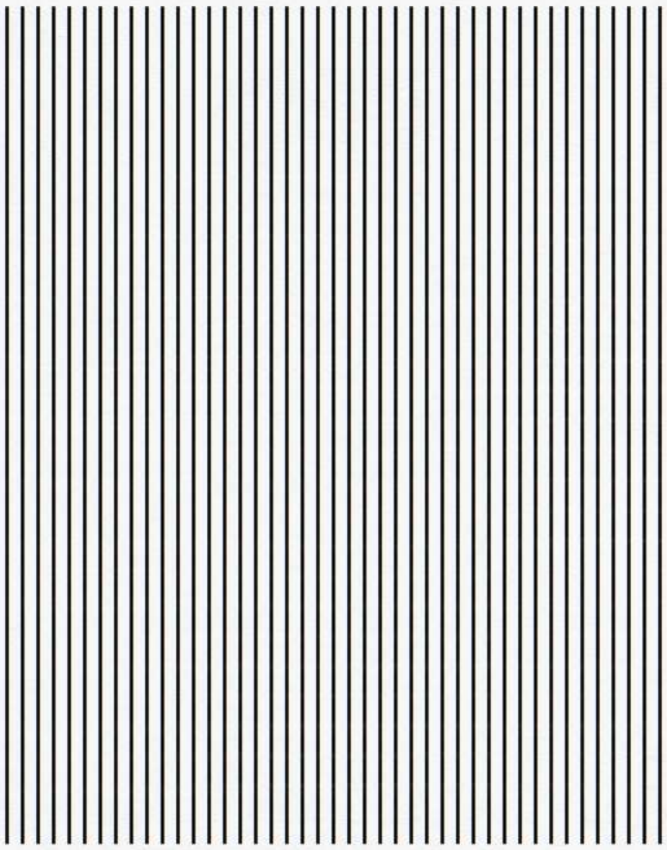
User manual



MODULUS

A ---------- PREPARIAMOCI AD USARE LA BASE MODULUS -

A.1: MODULUS e la sua BASE

MODULUS e´ un personal robot che cresce modularmente e la base e´ la sua struttura portante.

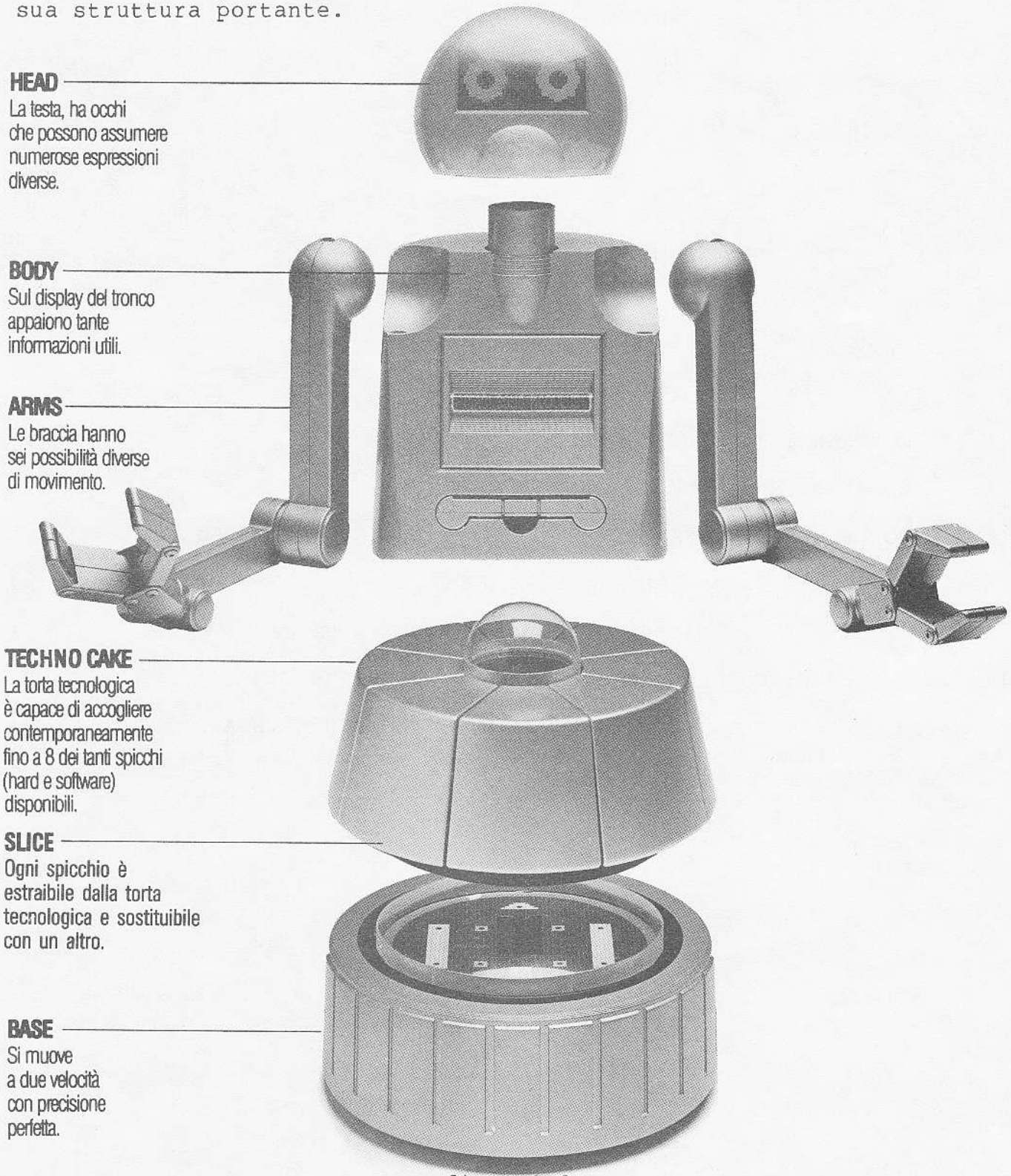


fig a.1.1

La BASE e´ prima di tutto il preciso organo di locomozione del robot MODULUS.

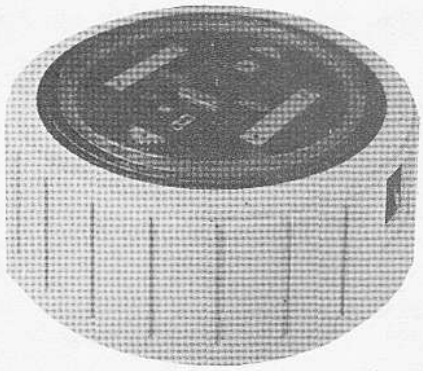


fig a.1.2

E´ dotata di due motori con cambio di velocita´ meccanico, controllati in velocita´ e posizione da un sistema a microprocessore dedicato a questa funzione.
Tutto cio´ permette di raggiungere la posizione voluta con una precisione fino ad una parte su mille.

Sulla parte superiore della BASE una serie di indicatori luminosi colorati informano ad ogni istante sulla situazione delle batterie e sul movimento in corso.

La BASE e´ circondata da una serie di sensori che permettono a Modulus di accorgersi di eventuali urti.
Da sola, la BASE MODULUS e´ una periferica di un qualsiasi home o personal computer: il collegamento con il computer di controllo puo´ essere fatto via cavo o via radio.

La versione via cavo (CABLE LINKED) e´ prevista per essere collegata ad un Commodore 64/128 e non e´ dotata di batterie proprie.
Quella via radio (RADIO FREQUENCY LINKED) dispone di di un apparecchio di comunicazione a radiofrequenza e di una tastiera (RF-COMMAND KEYBOARD) che consentono il collegamento senza fili a tutti gli home e personal computer dotati di interfaccia seriale RS 232 oppure il controllo diretto della BASE da tastiera.
In questo caso la BASE e´ anche dotata di batterie ricaricabili e non e´ piu´ vincolata fisicamente al computer di controllo.

Si puo´ poi inserire nella BASE, tramite un supporto (SUPPORT-ATTACHMENT), un meccanismo per disegnare con dei normali pennarelli (PLOTTER-DRAWING MECHANISM) oppure un aspirapolvere (VACUUM CLEANER) per piccole operazioni di pulizia.

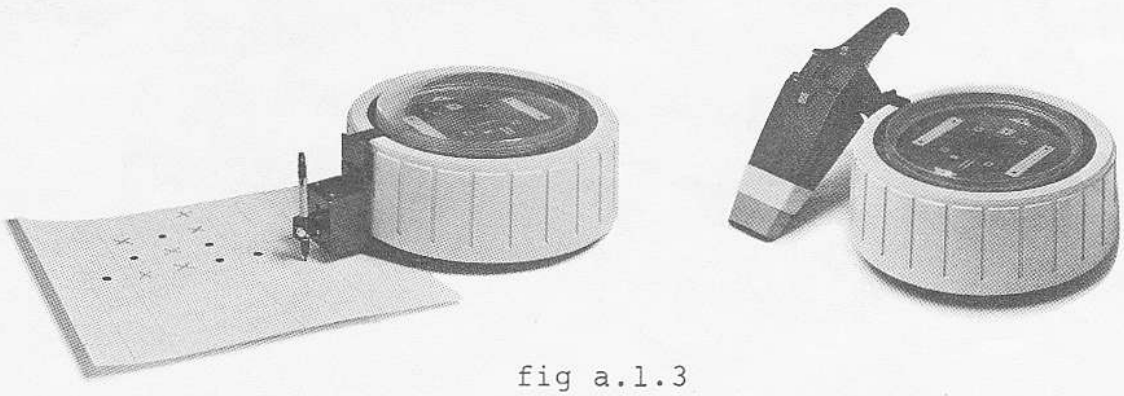


fig a.1.3

Fino a qui la BASE.
MODULUS puo´ poi crescere dotandosi di un supporto con 8 spicchi estraibili (TECHNO-CAKE o "spicchioteca") che possono ospitare le piu´ sofisticate funzioni: per cominciare uno spicchio "sistema di sicurezza" (HOME SECURITY) in grado di rilevare presenza di gas, acqua, fumo e di corpi in movimento. Tutto cio´ permette di prevenire sitazioni pericolose e l´accesso di persone indesiderate nella vostra casa.
MODULUS diventa cosi´ SERVICE & SECURITY ROBOT.

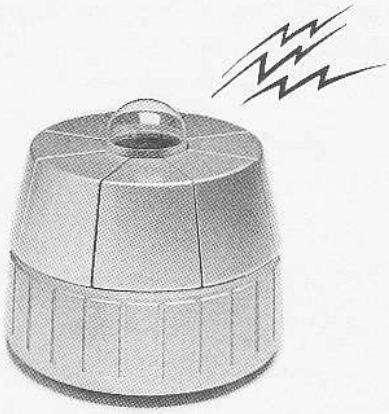


fig a.1.4

Ecco alcune fra le funzioni disponibili in spicchio: sintetizzatore vocale, stazione meteorolgica, sonar & navigazione, riconoscitore di presenza umana, riconoscitore di sorgenti sonore, CPU a 16 bit con 128K RAM e 128K ROM con floppy disk drive 3 1/2 " compatibile MS-DOS.

Su questo sistema si puo´ anche inserire un braccio molto sofisticato, dotato di 6 gradi di liberta´ e controllato da un sistema multi-microprecessore. La mano e´ in grado di portare oggetti pesanti fino a 500 grammi.

MODULUS, nella sua forma piu´ evoluta, e´ dotato anche di corpo con display alfanumerico, testa in grado di muoversi e assumere differenti espressioni e due braccia indipendenti: abbiamo cosi´ un robot umanoide (HUMAN ROBOT o MODDY).

MODDY viene gia´ dotato degli spicchi voce e CPU 16 bit, ma ognuno puo´ aggiungere altre funzioni inserendo i relativi spicchi nella spicchioteca.

La spicchioteca e´ in realta´ il vero sistema nervoso di MODULUS che mette in contatto tutti i microprocessori dedicati alle varie funzioni con la CPU a 16 bit e, se lo desiderate, con il vostro computer esterno.

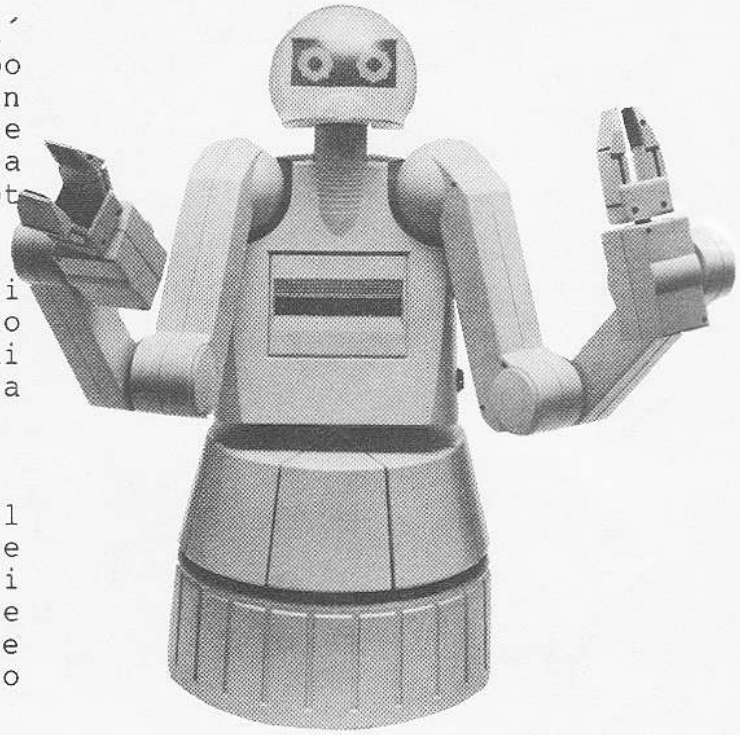


fig a.1.5

A.2: Come usare questo manuale

Anche questo manuale, come MODULUS, e´ modulare.
Ogni parte di MODULUS che acquisterete sara´ accompagnata da un fascicolo, che aggiungerete al raccoglitore del manuale, sottoforma di un nuovo capitolo.

Questo "modulo" del manuale, accompagnato alla BASE, spiega:

* come preparare la BASE a funzionare (capitolo A)

* come far muovere alla BASE i suoi primi passi (capitolo B)

* come controllare e programmare il movimento (capitolo C)

* come funziona la comunicazione con la BASE (capitolo D)

* come cavarsela con l´hardware (capitolo F)

* cosa leggere per fare un uso piu´ sofisticato del robot (capitolo G)

E se avete acquistato qualche accessorio della BASE:

* come montare ed usare la penna-plotter (capitolo H) e l´aspirapolvere (capitolo I)

Come usarlo?

** se intendete usare la BASE con un programma applicativo SIRIUS, allora l´unica lettura essenziale e´ quella dei capitoli A e B (anche H e I se usate gli accessori)

** se invece volete imparare a muovere la base con dei vostri programmi leggete anche i capitoli C, D ed E

** se avete acquistato una BASE controllata via "filo" da un Commodore 64/128 (BS 101), allora potete saltare i paragrafi A.4.2, A.4.3, B.2.1, B.2.2, D3

** se avete acquistato invece una BASE con telecomando RF (BS 101/RF), allora potete saltare i paragrafi A.4.1, B.3, D.2

** se avete acquistato il supporto con il plotter o l´aspirapolvere allora leggete i capitoli H ed I

** Infine se volete cavartela da soli per alcuni problemi elettronici leggete il capitolo F.
Se volete anche saperne di piu´ sull´argomento robot, per sfruttare appieno le caratteristiche di MODULUS, leggete il capitolo G

A.3: POWER SUPPLY & BATTERY CHARGER: l´energia per MODULUS

Il POWER SUPPLY & BATTERY CHARGER (che chiameremo piu´ brevemente PS & BC) e´ il mezzo con cui dare energia alla BASE MODULUS.

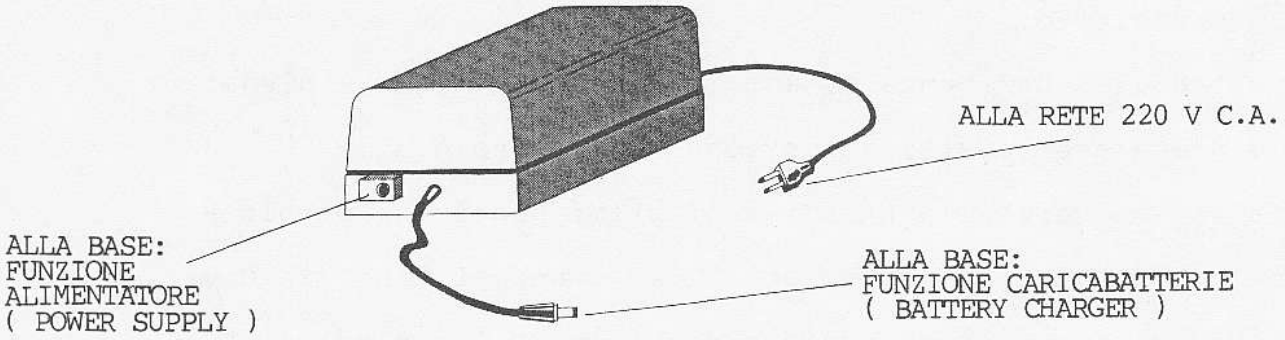


fig a.3.1

Del PS & BC si fanno due usi diversi a seconda della versione di BASE di cui siete in possesso; vediamo separatamente i due casi.

1. BASE dotata di collegamento RF (BS 101/RF)

Se avete acquistato questa configurazione controllate di avere anche il telecomando RF (RF COMMAND KEYBOARD) di cui parleremo nel paragrafo A.4.2 .
In questo caso la BASE contiene delle batterie ricaricabili: il PS & BC funziona da caricabatterie.
La BASE vi segnala automaticamente quando occorre caricare le batterie: guardiamo la base dall´alto.

Accendete la BASE, agendo sull´interruttore posto sul fianco, fino a che non si illumini la spia verde di avvenuta accensione (vedi fig a.3.2).

Guardate ora le spie che stanno alla sinistra del simbolo di batteria (vedi fig a.3.2); una delle tre si accendera´ con il seguente codice:

VERDE = BATTERIE CARICHE

GIALLO = BATTERIE QUASI CARICHE

ROSSO = BATTERIE SCARICHE

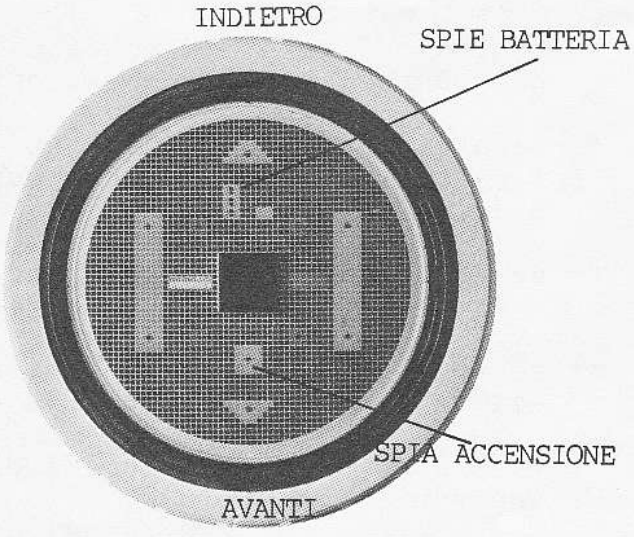


fig a.3.2

Se le batterie sono scariche occorle caricarle; procedere cosi´:

A) spegnete la BASE agendo sull´interruttore gia´ visto.

B) prendete il cavo di caricabatteria del PS & BC (vedi fig a.3.1 nella pagina precedente) e inseritelo nella corrispondente presa della BASE (fig a.3.3).
Lo scomparto prese e´ sul fianco in posizione diametralmente opposta all´interruttore.

C) inserite la spina C.A. del PS & BC in una presa di rete a 220 v C.A. .

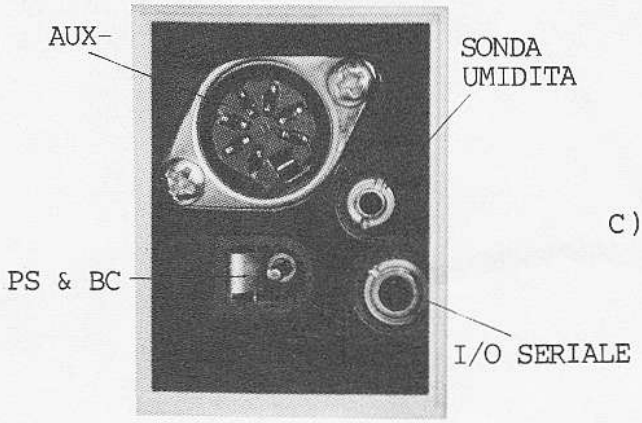


fig a.3.3

La carica delle batterie viene regolata automaticamente e dura al massimo 8 ore partendo da batterie completamente scariche.
Per verificare la carica accendete la BASE e controllate le spie di batteria: se la spia accesa non e´ la verde, spegnete la BASE e continuate la carica.

Se invece e´ accesa la sola spia verde, le batterie sono cariche per almeno il 70%, disinserite quindi lo spinotto del PS & BC dalla BASE e riaccendetela: ora la BASE MODULUS e´ pronta a funzionare.

2. BASE dotata di collegamento via cavo (BS 101)

In questo caso la BASE non contiene batterie proprie ed il PS & BC funziona da sorgente di energia elettrica a bassa tensione per la BASE MODULUS.
Si usa il lungo cavo (DATA & POWER CABLE) in dotazione.

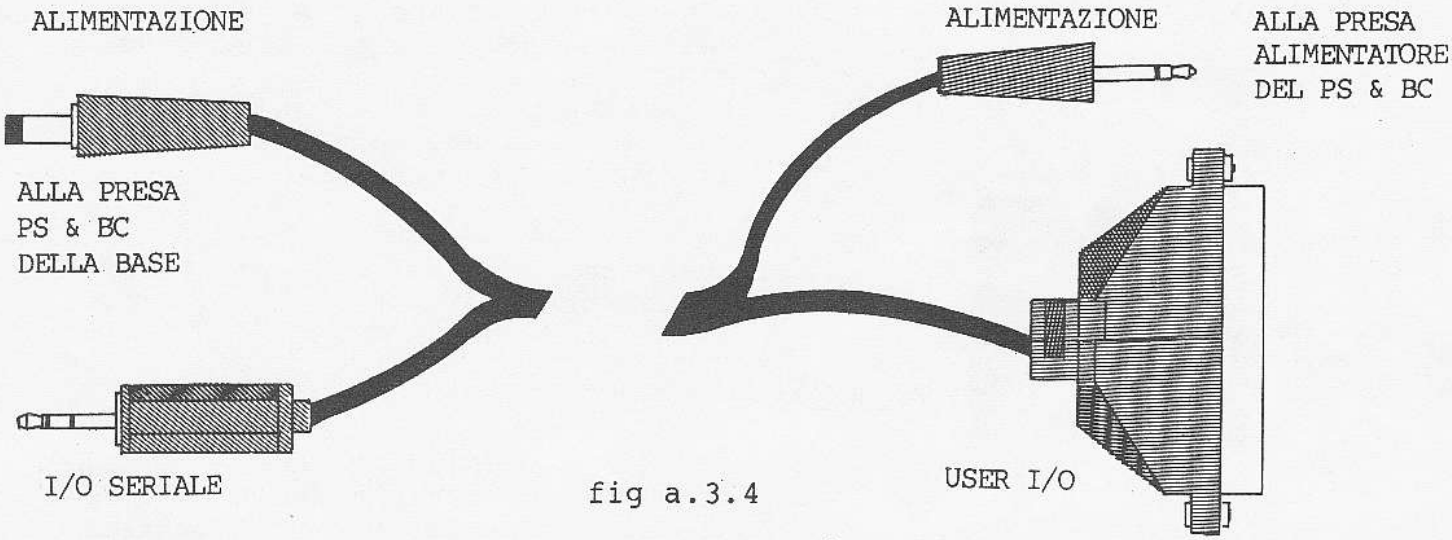


fig a.3.4

Questo cavo convoglia alla BASE sia l´energia elettrica che i comandi provenienti dal vostro Commodore.

Connettete lo spinotto di ALIMENTAZIONE alla apposita presa sul fianco della BASE (fig a.3.5).

Connettete l´altro spinotto di ALIMENTAZIONE alla presa sul PS & BC (fig a.3.1) .

Collegate la spina C.A. del PS & BC ad una presa di rete a 220 v C.A. e accendete la BASE, agendo sull´interruttore posto sul fianco opposto alle prese.
La spia verde illuminata indica che la BASE e´ ACCESA (fig a.3.5).

L´indicatore di carica batteria (fig a.3.5) dovra´ avere una spia verde accesa: ora la BASE MODULUS e´ pronta a funzionare.

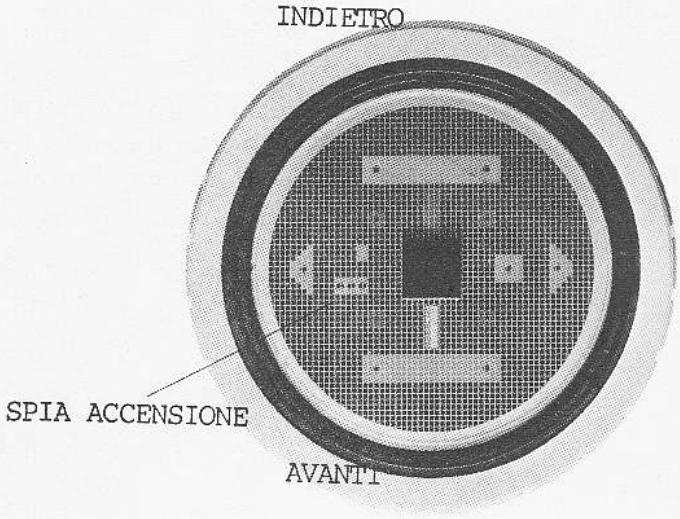


fig a.3.5

A.4.1: Il "filo diretto" con il Commodore 64/128

La BASE MODULUS e´, da sola, una periferica di un computer. Questo significa che e´ il computer che deve prendere l´iniziativa e comandare alla BASE di eseguire qualcosa.
Per noi questo "qualcosa" sara´ principalmente una serie di movimenti. Perche´ ci sia controllo del computer sulla BASE occorre che ci sia un canale di comunicazione.
Questo canale verso la BASE MODULUS puo´ essere realizzato fisicamente via cavo o via radio: in questo paragrafo ci occupiamo del collegamento via cavo.

Il collegamento via cavo e´ previsto esclusivamente con un computer Commodore 64 o 128 (per questioni di compatibilita´ elettrica; gli interessati possono leggere il paragrafo D.2).
Il cavo in questione e´ il DATA & POWER CABLE che e´ lungo circa 5 metri.

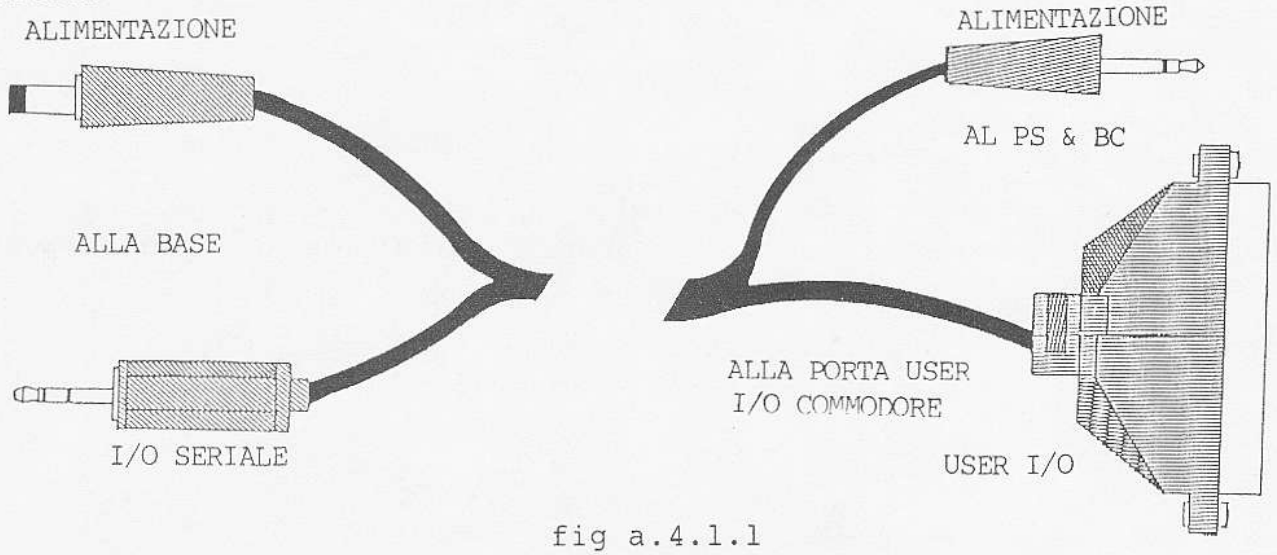


fig a.4.1.1

Questo cavo convoglia alla BASE sia l´energia elettrica che i comandi provenienti dal vostro Commodore.

Dovreste aver gia´ connesso i due spinotti di questo cavo relativi all´alimentazione elettrica della BASE (par A.3).

Completiamo ora il collegamento.

Se la BASE e´ accesa, spegnetela prima di effettuare le connessioni.

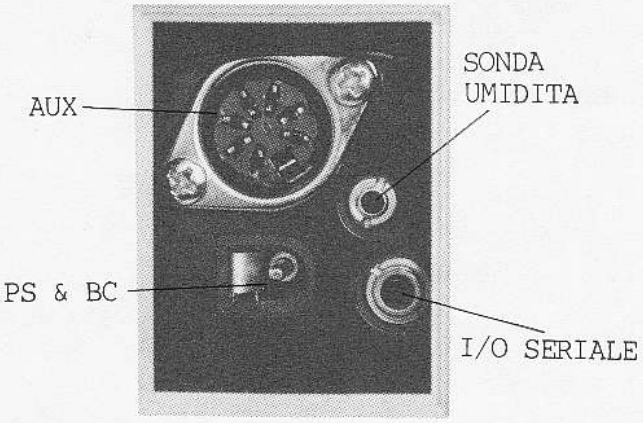


fig a.4.1.2

Connettete lo spinotto di I/O seriale alla apposita presa sul fianco della BASE (fig a.4.1.2).
Collegate il connettore a pettine USER I/O alla presa USER I/O del Commodore.
Notate che, malgrado le apparenze, vi e´ un solo verso con cui il connettore entra nella presa del computer e quindi non possono sorgere problemi.

Controllate che il lungo cavo steso sia libero di scorrere senza impigliarsi nel robot o in altri ostacoli.

La BASE MODULUS e´ ora pronta per le prime prove.

A.4.2: R.F. COMMAND KEYBOARD: la connessione via etere

La BASE MODULUS e´, da sola, una periferica di un computer. Questo significa che e´ il computer che deve prendere l´iniziativa e comandare alla BASE di eseguire qualcosa.
Per noi questo "qualcosa" sara´ principalmente una serie di movimenti. Perche´ ci sia controllo del computer sulla BASE occorre che ci sia un canale di comunicazione.
Questo canale verso la BASE MODULUS puo´ essere realizzato fisicamente via cavo o via radio: in questo paragrafo ci occupiamo del collegamento via radio.

Per rendere piu´ "aperto" e flessibile il nostro sistema abbiamo dotato il telecomando di una tastiera e di una intelligenza locale con un microprocessore dedicato al controllo diretto della BASE. Questo apparecchio si chiama RF-COMMAND KEYBOARD, o piu´ velocemente RF-CK.
Il canale di comunicazione radio puo´ quindi essere usato in piu´ modi operativi:

MODO 1. per comandare la BASE direttamente usando solo la tastiera dell´RF-CK

MODO 2. per comandare la BASE con un home o personal computer, rendendo la tastiera dell´RF-CK "visibile" anche dal computer

MODO 3. con configurazioni piu´ sofisticate di MODULUS (CPU a 16 bit a bordo) e´ il robot stesso che "legge" la tastiera e la usa per acquisire informazioni ddall´utente; se il computer e´ connesso alla RF-CK, allora la CPU di MODULUS puo´ colloquiare direttamente con il computer stesso

In questo manuale, che riguarda solo la BASE MODULUS, ci occuperemo solo dei modi 1 e 2.
Cominciamo vedendo come si presenta la RF-CK:

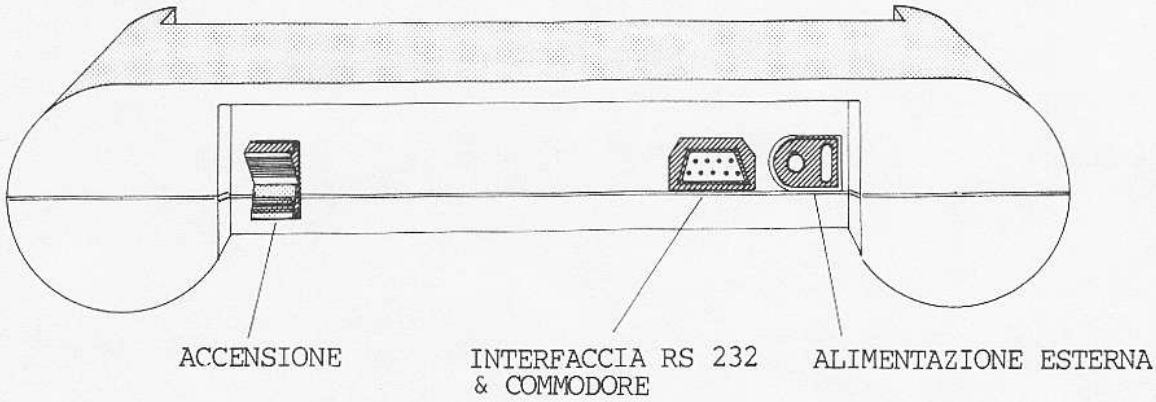


fig a.4.2.1

La RF-CK viene alimentata da 8 pile a stilo da 1.5 v .

Alternativamente si puo´ alimentare la RF-CK con un alimentatore stabilizzato a 12v C.C. da 0,3 A, tramite la presa coassiale che si vede in fig a.4.2.1 (il negativo e´ collegato al centro).

Se si usano le pile, la prima operazione da fare e´ quella di inserirle all´interno della RF-CK.
Per fare cio´ occorre aprire la RF-CK: tenete la tastiera rivolta verso il basso.
I portapile sono alloggiati nelle due conche laterali.
Sfilate gli sportellini ad incastro sulle conche ed inserite le pile.
Messe le pile, richiudete gli sportellini e sistemiamo la RF-CK con la tastiera verso l´alto.

Cercate di tenete la RF-CK in modo che l´antenna rimanga piu´ in verticale possibile.

Attenzione: questa e´ un´operazione semplice ma importante per non diminuire la portata di trasmissione!

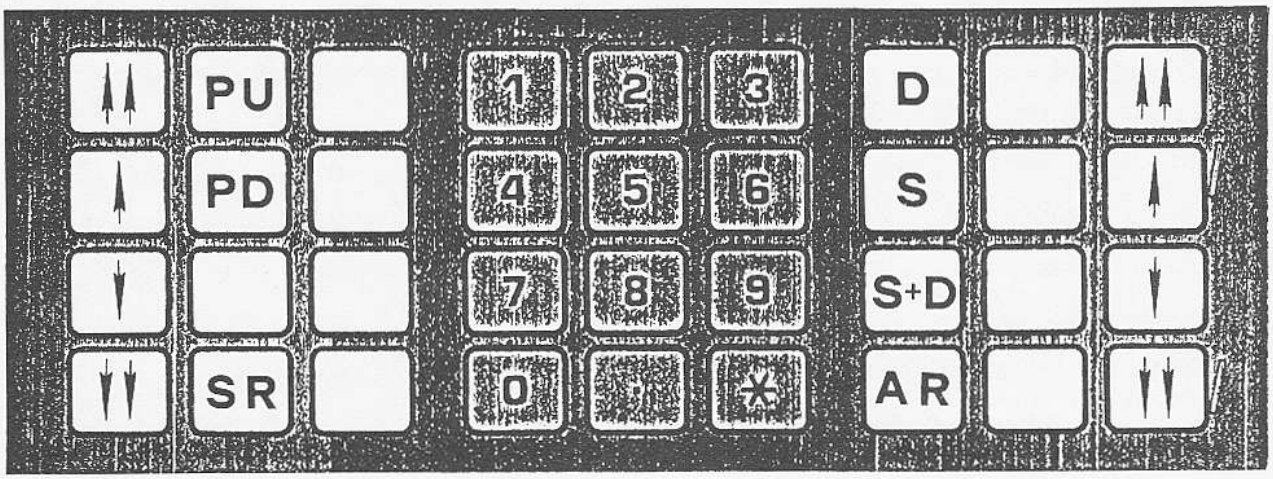


fig a.4.2.2

Ora siamo pronti per accendere l´RF-CK.

Siccome faremo la prima conoscenza con il movimento della BASE MODULUS usando il canale di comunicazione in MODO 1 (cioe´ per comandare la BASE direttamente usando solo la tastiera dell´RF-CK), occorre comunicare questa decisione all´RF-CK.
Vediamo come si procede.

SELEZIONE MODO 1

E´ sufficiente che teniate premuto un tasto qualsiasi della tastiera mentre si accende l´RF-CK.

L´RF-CK si accende agendo sull´interruttore (vedi fig a.4.2.1) fino a che non si illumina la spia rossa di "acceso", sopra la tastiera.
L´RF-CK segnalera´ di aver accettato il modo operativo 1 emettendo tre sonori "bip".
Quando rilasciate il tasto, l´RF-CK e´ pronto a funzionare.

Per fare i primi movimenti vi consigliamo di usare il capitolo B.
Potete quindi spegnere per il momento l´RF-CK agendo sull´interruttore.
Quando lo riaccenderete, selezionerete ancora il modo 1 ripetendo la procedura appena vista.

Il paragrafo successivo (A.4.3) riguarda la connessione del vostro home/personal computer alla RF-CK (per poter usare il MODO 2: controllo BASE da computer esterno).
Nelle prime prove conviene usare solo il modo 1 e noi lo useremo di fatto nel capitolo B. Potete quindi rimandare la lettura del paragrafo A.4.3 a dopo quella dell´intero capitolo B.

A.4.3: Il vostro computer e la R.F. COMMAND KEYBOARD

Per far comunicare il vostro computer con la BASE MODULUS usando il canale radio dell´RF-CK bisogna collegarlo alla RF-CK stessa.
Cio´ implica l´uso dell´RF-CK nel MODO 2, cioe´ controllo della BASE attraverso il vostro home/personal computer; vediamo come selezionare questo modo operativo.

SELEZIONE MODO 2

Se all´atto dell´accensione dell´RF-CK non si tiene premuto alcun tasto della tastiera viene automaticamente selezionato il modo 2.
Aggiungeremo alcuni dettagli sull´argomento nelle pagine successive.

Il modo 2 prevede che il vostro computer sia collegato alla RF-CK.
La cosa e´ possibile se il vostro computer e´ dotato di interfaccia seriale tipo RS 232. Se avete un Commodore 64 o 128 l´interfaccia seriale e´ gia´ contenuta nel computer (ed e´ una variante del tipo RS 232) .
L´RF-CK accetta la connessione sia con computer dotati di interfaccia RS 232 che con i Commodore 64/128 con la RS 232 "variante Commodore": occorre pero´ utilizzare dei cavi differenti.
Nel richiederli dovete specificare se usate un Commodore 64 o 128 oppure un altro computer con interfaccia RS 232.
Se invece intendete fabbricarvi tali connettori allora consultate il paragrafo F.1 (e se ne volete sapere di piu´ leggete anche il paragrafo D.3).

In tutti e due i casi il cavo si inserisce nel connettore a vaschetta da 9 poli (fig a.4.3.1) dell´RF-CK.

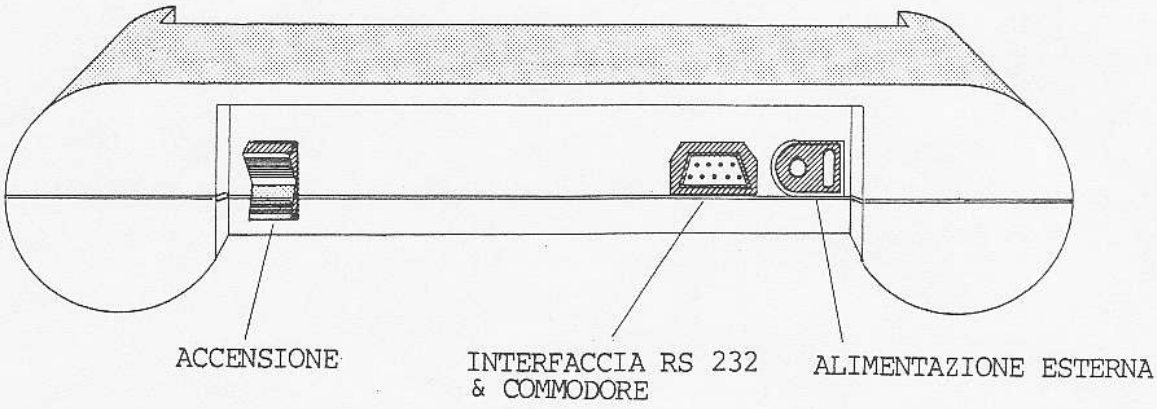


fig a.4.3.1

Spegnete l´RF-CK ed inserite bene a fondo il corrispondente connettore del cavo nella presa dell´RF-CK.
Questa operazione e´ molto importante dal momento che la connessione deve essere stabilissima per assicurare un buon colloquio fra il vostro computer e la BASE MODULUS.

Collegate l´altro connettore del vostro cavo nella corrispondente presa a pannello dell´interfaccia RS 232 del vostro computer, avvitando le viti di serraggio eventualmente presenti, o nella presa USER I/O del vostro Commodore 64/128.

ATTENZIONE: ESEGUITE QUESTE CONNESSIONI SEMPRE CON IL COMPUTER E LA RF-CK SPENTI. ALTRIMEMTI I DUE APPARECCHI POTREBBERO RIPORTARE DANNI.

L´interfaccia seriale va programmata: la RF-CK deve solo sapere la velocita´ di scambio dati con il vostro computer.
Le velocita´ possibili sono due: 300 o 9600 baud.
Se usate un Commodore 64 o 128 la velocita´ deve essere 300 baud (perche´ nel CBM 64 e 128 non e´ prevista la velocita´ di 9600 baud).

ATTENZIONE: la RF-CK esce dalla fabbrica gia´ programmata per colloquiare a velocita´ di 300 baud.
Se intendete usare tale velocita´, come nel caso Commodore, allora non occorre seguire le istruzioni ai punti 1, 2 e 3: saltate direttamente al punto 4.

Per selezionare la velocita´ sull´RF-CK occorre procedere cosi´:

1. aprite la RF-CK tenendo la tastiera verso il basso, sfilando gli sportellini sopra le conche e svitando le 4 viti, sotto i portapile, che tengono insieme i due gusci che compongono il contenitore plastico.
   Vediamo qui sotto come si presenta la situazione.

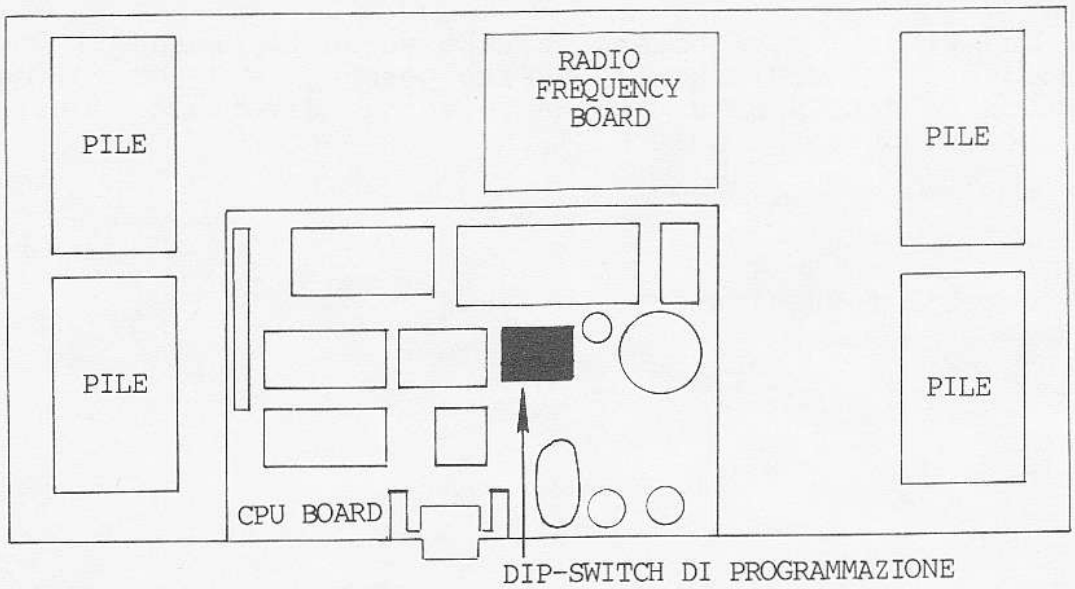


fig a.4.3.2

2. localizzate il dip-switch di programmazione usando la figura sopra.

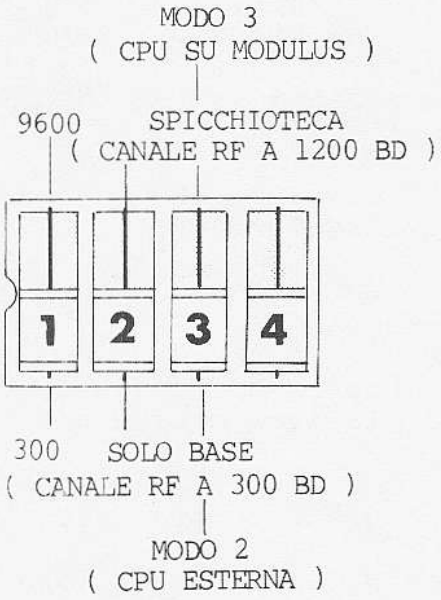


fig a.4.3.3

Se volete selezionare la velocita´ di 300 baud, bisogna spostare verso il basso lo switch numero 1 (fig a.4.3.3).
Viceversa se intendete selezionare la velocita´ di 9600 baud portate lo switch 1 verso l´alto.

Lo switch 3 controlla la selezione fra modo operativo 2 e 3.
Questo perche´ anche il modo 3 (che riguarda pero´ solo configurazioni di MODULUS piu´ complesse che non la sola BASE) verra´ selezionato accendendo la RF-CK senza contemporaneamente premere un tasto, come per il modo 2.
La RF-CK assume il modo di funzionamento 2 o 3 a seconda di come e´ posizionato lo switch 3.
Normalmente la RF-CK esce dalla fabbrica con lo switch programmato su modo 2.
In ogni caso lo switch 3 deve avere la levetta in basso per selezionare il modo di funzionamento 2, cioe´ il controllo della BASE MODULUS e´ effettuato dal computer eventualmente connesso alla RF-CK.

Anche lo switch 4 deve essere portato verso il basso (= MODULUS non dotato di SPICCHIOTECA). Questo perche´ la RF-CK colloquia via radio con la BASE MODULUS ad una velocita´ diversa da quella usata con MODULUS dotato di SPICCHIOTECA.

3. richiudete il contenitore della RF-CK.

4. ora occorre programmare l´interfaccia seriale del vostro computer; i parametri da programmare sono:

   * VELOCITA´ = 300 o 9600 baud (a seconda della vostra scelta; se Commodore allora selezionare 300)

   * STOP BIT = 1

   * DATA BIT = 8

   * PARITA´ = no

   * se si opera a 9600 baud handshake busy/free effettuato con i segnali di controllo RTS e CTS (gli altri segnali di controllo modem non vengono gestiti

   * se si opera a 300 baud niente handshake: interfaccia a 3 fili (3 wire)

Se avete gia´ esperienza con le interfacce RS 232 e la loro programmazione potrete fare tutto con questi dati.

Se invece non vi dicono molto, leggete il capitolo del manuale del vostro computer che descrive l´uso e la programmazione dell´interfaccia RS 232; per il Commodore 64, ad esempio, e´ nel capitolo 6 della Guida di Riferimento al Programmatore.
Poi leggete il paragrafo D.3 di questo manuale.

B ---------- CONTROLLO DELLA BASE: PRIME OPERAZIONI -

B.1: Guida al movimento della BASE (livello 1)

La BASE MODULUS e´ principalmente un preciso sistema di movimento: prestazione che lo distingue dalla precedente generazione di personal robot.

Programmare un movimento di precisione e´ un problema che puo´ essere semplice o complicatissimo, a seconda delle vostre esigenze e della vostra curiosita´.
Nel corso di tutto il capitolo B affronteremo insieme il problema nel modo piu´ semplice possibilie.
Nonostante la semplicita´, questo sistema assicura di poter portare il vostro MODULUS in ogni posizione voluta.

Nel capitolo C affinerete le vostre conoscenze e potrete sviluppare vostri sistemi per guidare il movimento della BASE MODULUS.

Il nostro robot e´ eqipaggiato con due motori che vengono controllati da un microprocessore dedicato a questo scopo.
I motori sono controllati in velocita´ e posizione (e vedremo che volendo si potra´ programmare anche l´accelerazione) da un sistema che minimizza gli errori di movimento, arrivando ad una precisione di fino 1 parte su 1000.

Ma vediamo di fare conoscenza con la nostra BASE MODULUS: capovolgiamola (a macchina spenta !!!) e lasciamola a "ruote all´aria".

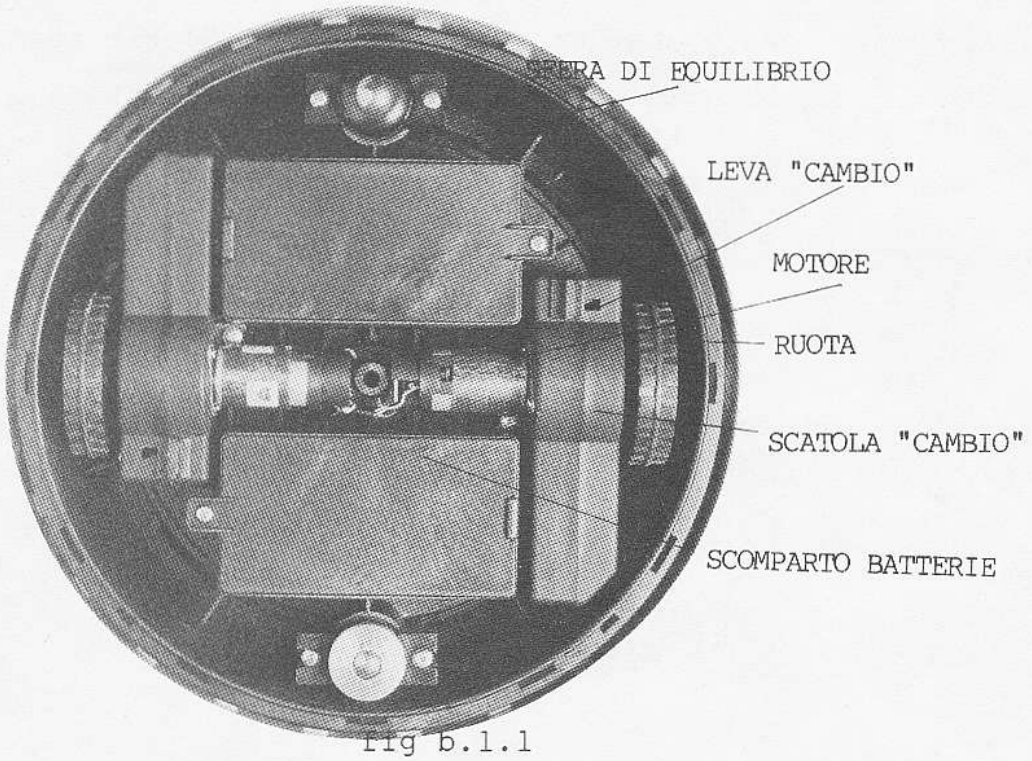


fig b.1.1

Rovesciando la BASE, la "gonnellina" laterale si appoggera´ a terra.

Notiamo subito le due grosse ruote.

Le ruote sono un particolare molto importante per un robot: uno slittamento costituirebbe una perdita di precisione nel movimento, visto che esso non possiede "occhi" per conoscere la propria posizione.
Immaginate di dovervi muovere ad occhi chiusi in un ambiente sconosciuto, con le orecchie tappate e con "le mani legate": potete solo contare i passi e sapete quanti passi fare per raggiungere la meta.
Se scivolate il conto dei passi non torna piu´: non potete piu´ sapere con certezza dove siete.

Diversamente dagli altri personal robot, le ruote di MODULUS sono state progettate con un disegno particolare e realizzate con materiali che minimizzano i problemi di slittamento.

Oltre a queste MODULUS possiede, nelle configurazioni piu´ evolute, dei "sensi" (inseribili nella spicchioteca - vedi par A.1) che permettono anche una verifica assoluta della posizione nell´ambiente.

Ma torniamo ad osservare la BASE.
Noterete una scatola nera attaccata ad ogni ruota.

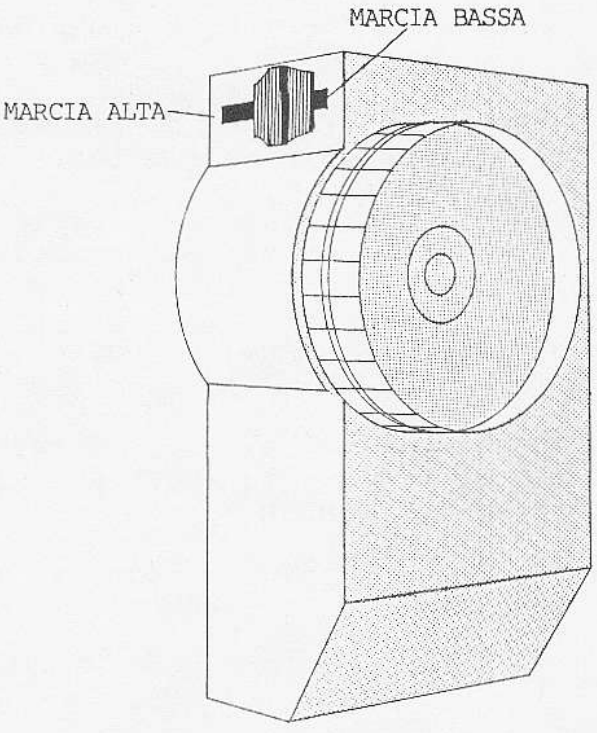


fig b.1.2

Questa contiene un meccanismo di demoltiplica che costituisce il "cambio" di MODULUS: potete infatti selezionare, per ogni ruota, due rapporti di demoltiplica o, se preferite, "marce": una bassa che assicura minori velocita´, ma maggiore forza nel movimento ed una alta, che assicura maggiori velocita´ ma minore forza nel movimento.

Il rapporto fra le due marce e´ di 1 a 4; vale a dire che con la marcia bassa, a parita´ di velocita´ che programmerete su MODULUS, otterrete velocita´ effettive 4 volte inferiori che nella marcia alta.

La marcia bassa si seleziona portando la leva, presente sulla scatola nera (fig b.1.2), verso l´esterno, cioe´ verso la ruota.
La marcia alta, viceversa, portandola verso l´interno.

Normalmente occorre che la selezione della marcia sia uguale sui due cambi, per ottenere un movimento controllabile facilmente.
Controllate ora che i due cambi siano sulla marcia bassa, cosa che assicura piu´ controllabilita´ manuale e maggior precisione.

Vedete poi due sfere di equilibrio (fig b.1.1) diametralmente opposte: lo scopo di queste sfere e´, come dice la parola stessa, di equilibrare la BASE, dandogli 4 appoggi.
Abbiamo scelto di usare sfere e non ruote per raggiungere una maggiore uniformita´ di comportamento in tutte le direzioni.

Se provate a toccare le sfere, noterete che rientrano: questo per evitare che eventuali piccoli ostacoli, o asperita´ del terreno, disturbino il movimento di MODULUS.

La sfera anteriore ha una sospensione piu´ cedevole di quella posteriore.
Inoltre, ruotando i supporti delle sfere in senso orario, si bloccano le sospensioni: questa prestazione serve solo quando MODULUS cresce da una configurazione all´altra, cambiando in peso ed in altezza; il sistema di regolazione e´ gia´ stato tarato per la BASE e non va toccato.

Quindi, se avete bloccato le sfere, sbloccatele ruotando i supporti in senso antiorario.
Ancora sul fondo vedete due sportellini che permettono l´accesso alle batterie ermetiche di MODULUS: anche questi non richiedono normalmente apertura, a meno di guasti, che verranno comunque gestiti dall´assistenza tecnica.

Precisiamo che, se avete acquistato la BASE in versione controllata via cavo (BS 101), non essendo questo modello alimentato da batterie, non troverete batterie dentro agli sportelli.
Sara´ cosi´ possibile sfruttarli, nel caso si verificassero eccessive oscillazioni nel moto della BASE, per stabilizzarla appesantendola: sara´ sufficiente piazzare dei pesi dentro agli sportellini citati.

Bene, ora che avete terminato l´ispezione della BASE a "testa in giu´", rimettetela con le ruote a terra.

Togliete per un momento la "gonnellina" e notate che al suo interno esiste un solo incastro che permette di bloccarla in corrisponeza della costina sporgente da un lato del corpo della BASE.

Rimontatela, facendo combaciare l´incastro e la costina suddetti in modo che la "gonnellina" non ruoti sul supporto.

Osserviamo finalmente la BASE dall´alto.

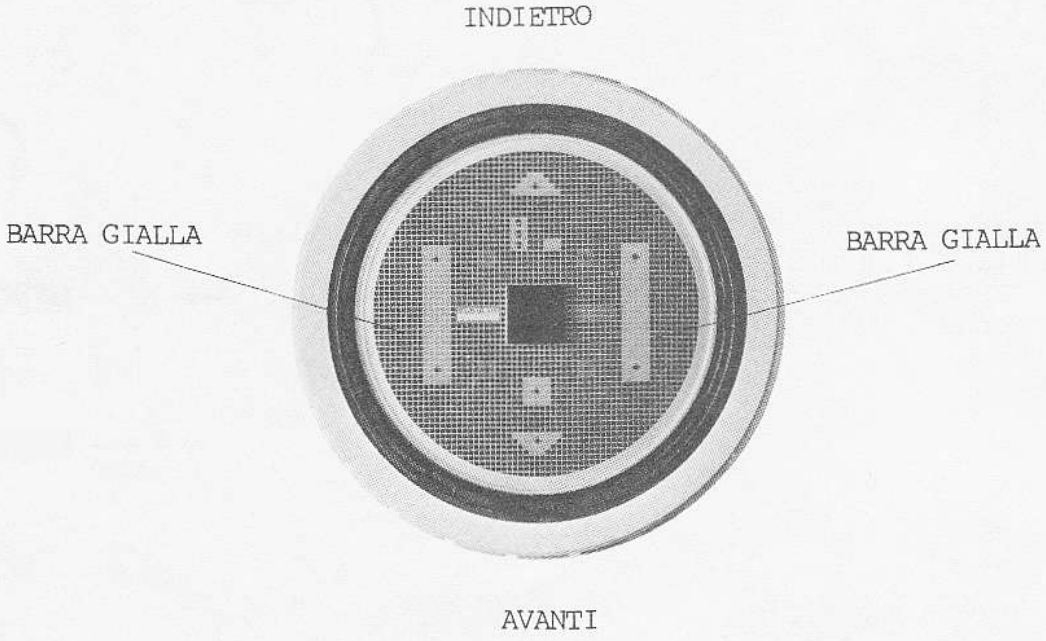
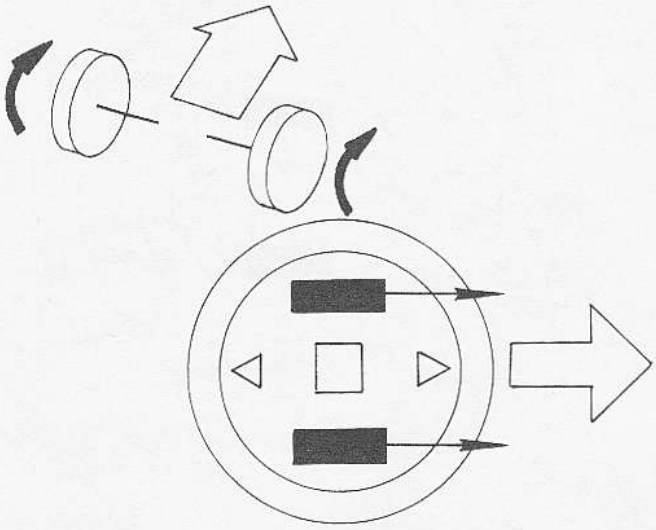


fig b.1.3

Le due barre gialle ricordano le due ruote viste dall´alto.
Le due ruote sono indipendenti: pensiamo come si puo´ combinare il loro movimento.



Potete ben immaginare che, se le ruote girano nello stesso senso e alla stessa velocita´, la BASE si sposta andando diritta.

fig b.1.4

Se le due ruote girano nello stesso senso, ma dall´altra parte, la BASE si muove sempre diritta ma nell´altro senso.

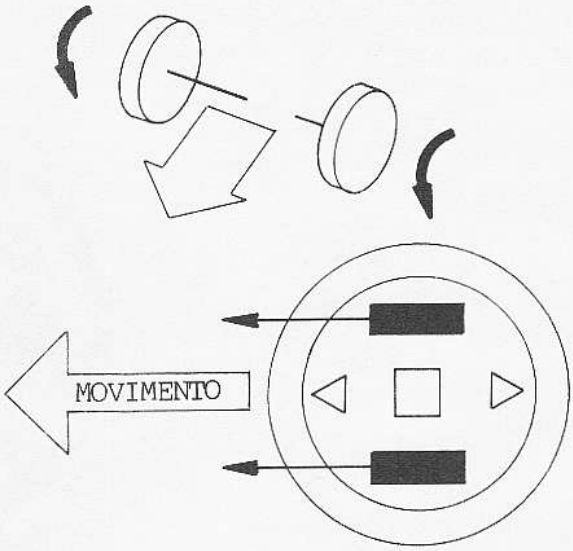


fig b.1.5

E se una ruota gira in un senso e l´altra nell´altro alla stessa velocita´?

Non dimenticate che le due ruote sono legate alla stessa carrozzeria e che la BASE non si puo´ ... dividere in due!
Alla fine si ottiene che la BASE ruota su se stessa.

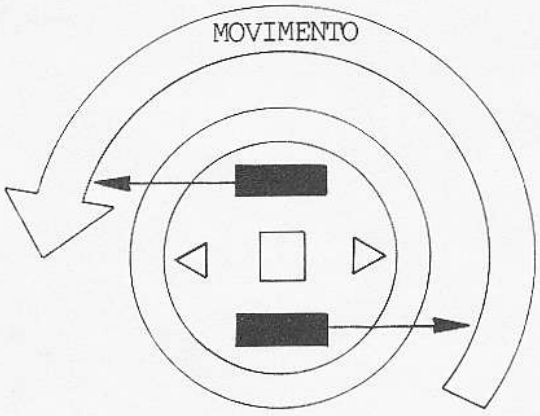
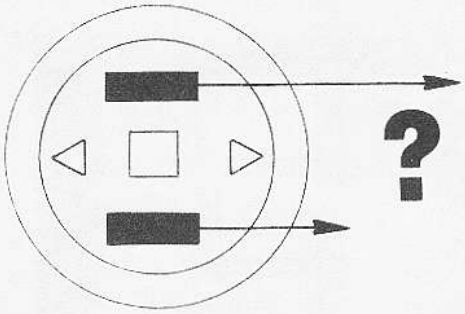


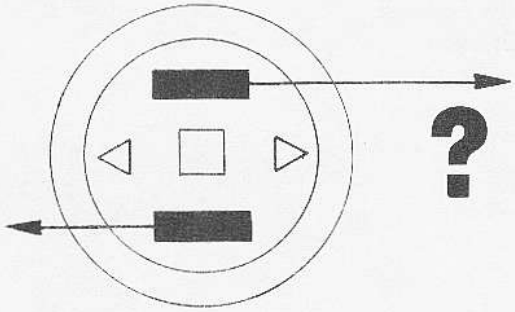
fig b.1.6

Fino a qui niente di difficile da immaginare.

Provate ora voi ad immaginare cosa succede se le due ruote girano a velocita´ diverse, prima nello stesso senso e poi, magari, in senso inverso.



Qui,ad esempio, se una ruota fa un giro in un secondo, l´altra ne fa due ... che succede?



E qui?

fig b.1.7

Se non ci riuscite o non avete voglia di spremervi le meningi, perche´ non chiederlo alla BASE MODULUS?

Vedremo come nei prossimi paragrafi.

B.2.1: R.F.COMMAND KEYBOARD: tastiera e funzioni

Questo paragrafo e´ riservato a chi possiede una BASE MODULUS con comando via radio; chi possiede la versione via cavo passi a leggere il paragrafo B.3 .

Riprendiamo ora in mano la RF-COMMAND KEYBOARD o RF-CK.
Osserviamone attentamente la tastiera.

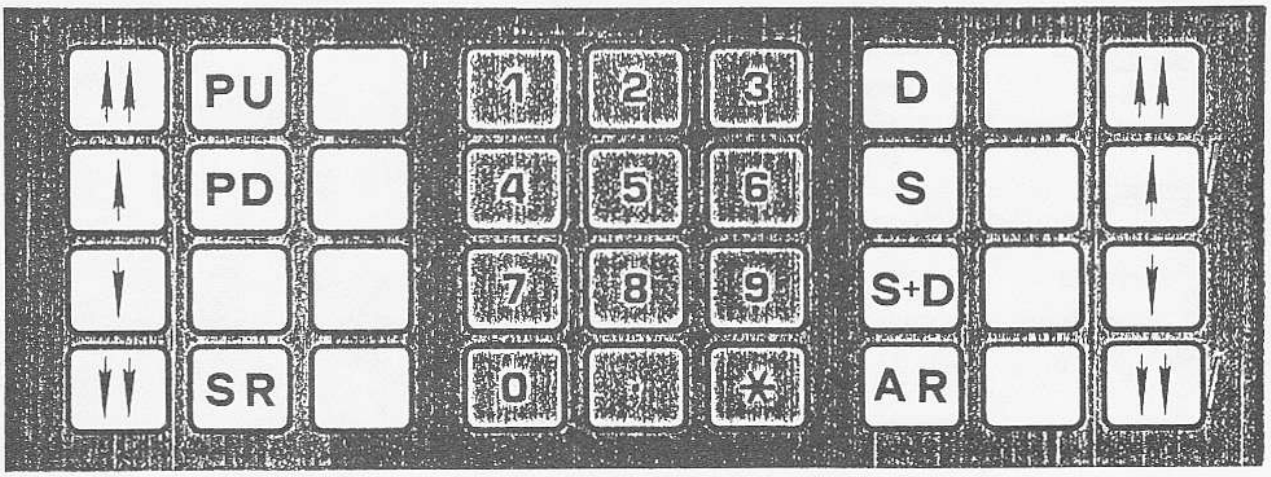


fig b.2.1.1

Al centro abbiamo una normale tastiera numerica, mentre sulla destra e sulla sinistra solo alcuni tasti riportano dei simboli.

Passiamo solo per un istante in rassegna i vari tasti.

SR : ci servira´, come vederemo fra poco, a far "dimenticare" alla BASE un urto.

PU e PD : riguardano il funzionamento della base con la penna-plotter o l´aspirapolvere: noi non li useremo per le prime prove. Se possedete questi accessori e non sapete resistere alla curiosita´ allora potete leggere subito il capitolo H o I.

⇈ : motore (destro se sulla parte destra della tastiera, sinistro se sulla parte sinistra) avanti veloce.

↑ : motore (destro o sinistro) avanti lento.

↓ : motore (destro o sinistro) indietro lento.

⇊ : motore (destro o sinistro) indietro veloce.

Questi 4 comandi a destra valgono per il motore di destra, a sinistra valgono per il motore di sinistra.

Ci sono poi altri quattro tasti (D, S, D+S, AR) che si usano solo con MODULUS dotato di spicchioteca e modulo di "service & security" e non hanno alcun effetto sulla BASE.
Ce ne occuperemo quindi nel relativo manuale.

Vediamo invece di capire perche´ tanti tasti sono in "bianco".

Abbiamo visto nel paragrafo A.4.2 che l´RF-CK puo´ funzionare in vari modi.

Se si opera nel modo 1 la tastiera serve a comandare direttamente il movimento della BASE MODULUS.

Se si opera nel modo 2 allora la tastiera comunica solamente al computer connesso alla RF-CK quale tasto e´ stato premuto: sara´ il computer stesso a decidere quali azioni prendere o far prendere alla BASE.

Lo stesso avviene se si opera nel modo 3, solo che invece e´ il computer a 16 bit a bordo della versione piu´ evoluta di MODULUS a ricevere l´informazione di tasto premuto.

Possiamo ora comprendere che nei modi 2 e 3 il significato che ha ogni singolo tasto non e´ fisso, ma viene deciso dal computer (o meglio dal programma che sta girando sul computer).
E´ per questo che SIRIUS fornisce anche delle mascherine non pre-stampate da applicare alla tastiera della RF-CK: potrete cosi´ usarle per segnare le funzioni che associerete ad ogni tasto.

Invece nel modo 1, che useremo ora, la tastiera invia direttamente alla BASE MODULUS dei comandi ben definiti e fissati.

Ora teniamo l´antenna piu´ verticale possibile, accendiamo l´RF-CK, tenendo premuto un tasto qualsiasi per selezionare il modo 1, attendiamo i 3 bip di conferma e ... passiamo al paragrafo successivo!

B.2.2: Movimento della BASE con R.F. COMMAND KEYBOARD

Mettete la BASE al centro della stanza e lontana da ostacoli.

Tutto e´ pronto: la BASE MODULUS e´ accesa e l´RF-CK anche.
Se cosi´ non fosse accendeteli (non importa in che ordine).
Se non vi ricordate come farlo bene, allora rileggete i paragrafi A.3 e A.4.2 .

Prendete con le due mani l´RF-CK, tenendolo per i due lati.
Usate i pollici per comandare il movimento (fig b.2.2.1)

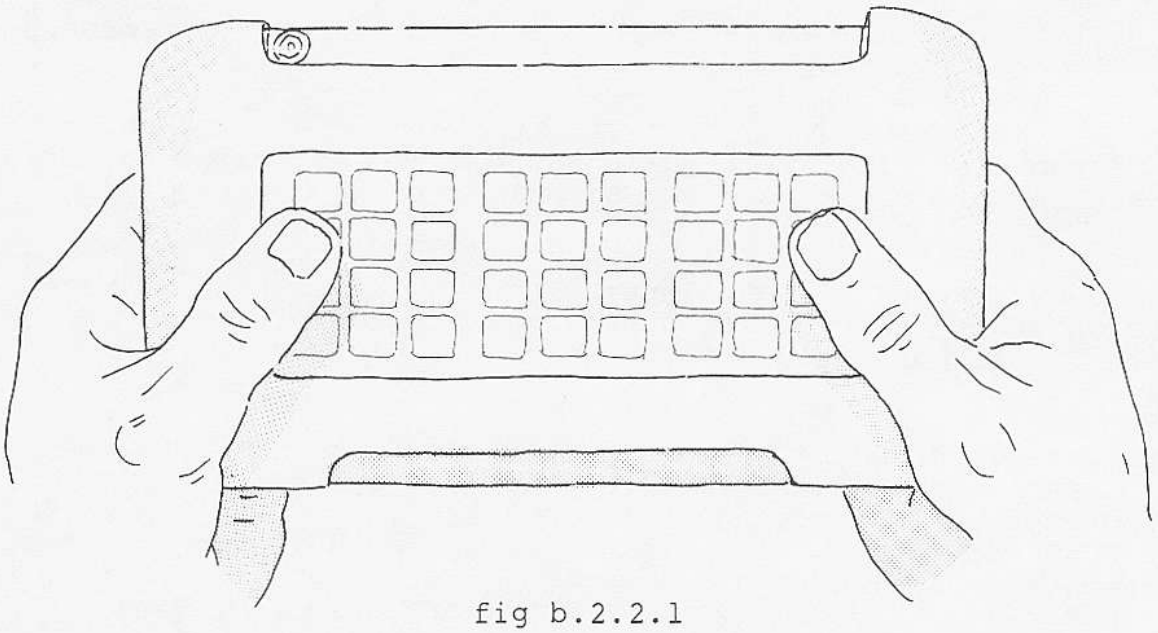
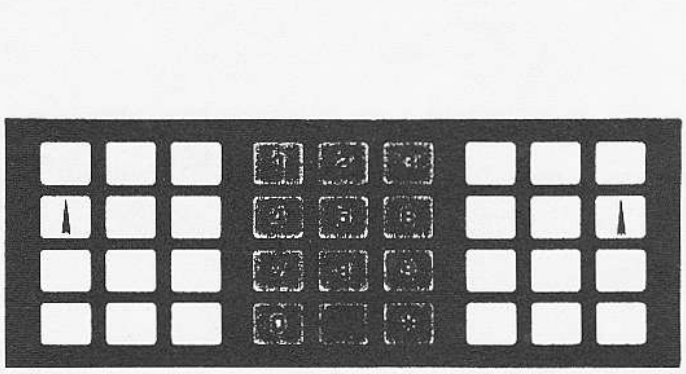


fig b.2.2.1

Premete insieme i tasti di avanti lento destro e sinistro:



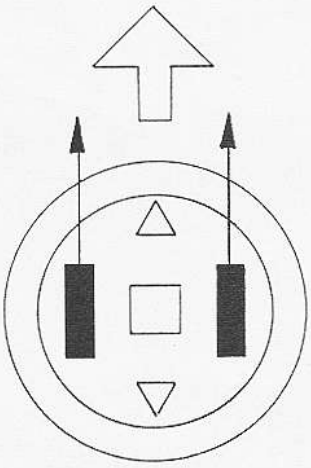


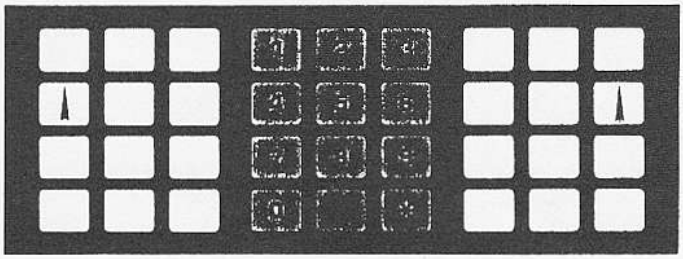
fig b.2.2.2

La BASE si muovera´ in avanti e continuera´ a muoversi fino a che tenete premuti i due tasti.

Provate piu´ volte.
Noterete che, quando si premono i tasti, l´RF-CK lo segnala con un "bip" e che quando la BASE si muove si accendono un po´ di spie luminose.

Non ci vuole molto a capire il significato delle spie che si accendono e per ora ve lo lasciamo indovinare: con poche prove lo capirete.
Ma se proprio volete saperlo subito potete leggere il paragrafo C.2 .

Bene: ora premete di nuovo insieme i tasti di avanti lento a destra e sinistra ...



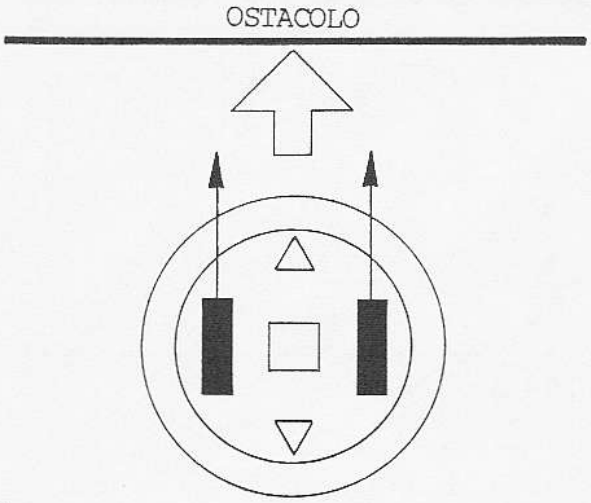


fig b.2.2.3

... e teneteli ben schiacciati fino a che la BASE MODULUS non urti contro una parete o un altro ostacolo.

Che succede?
La BASE si ferma automaticamente dopo l´urto.

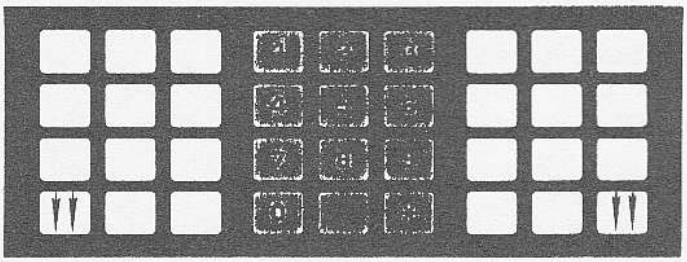
La BASE MODULUS e´ molto prudente e non vuole danneggiare ne´ se stessa ne´ gli altri.

Si sono poi accese delle spie rosse: la BASE ricorda cosi´ l´urto subito.

Per dimenticare l´urto la BASE MODULUS vuole proprio una vostra "autorizzazione ... scritta".

Premete a questo punto il tasto "SR" della RF-CK: la BASE ha ora ricevuto la vostra autorizzazione.

Premete ora i due tasti di indietro veloce insieme.



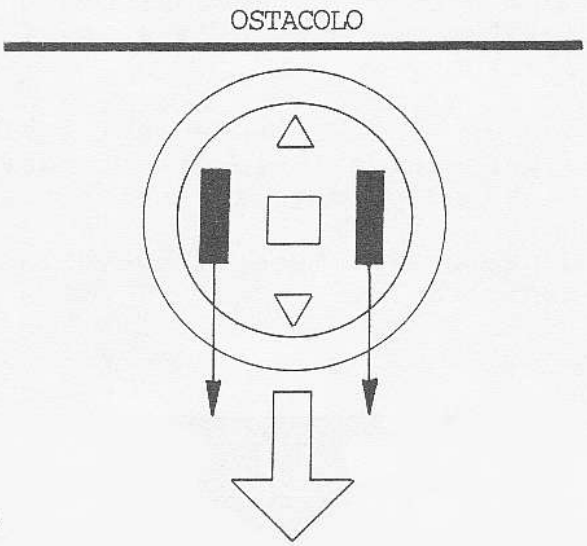


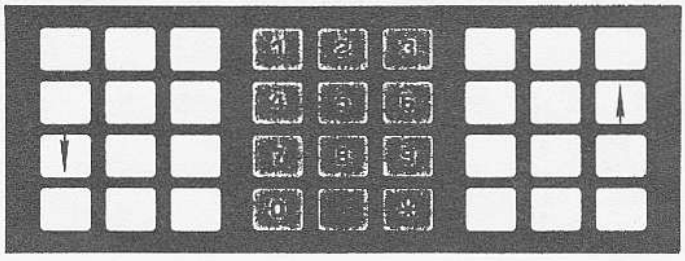
fig b.2.2.4

La base ritorna velocemente verso di voi: fermatela rilasciando i tasti.

Se la BASE non reagisce come dovrebbe, provate a premere il pulsante "SR" e poi procedete con i comandi voluti.

ATTENZIONE: la portata del telecomando e´ limitata per rientrare nelle norme concesse dalla legge.
La portata media per avere una comunicazione affidabile e´ di 20 metri.
Se ci sono muri o strutture metalliche frapposti fra la BASE e l´RF-CK la portata diminuisce.

Ora provate a far ruotare la BASE: tenete schiacciati il tasto avanti lento a destra ed il tasto indietro lento a sinistra.



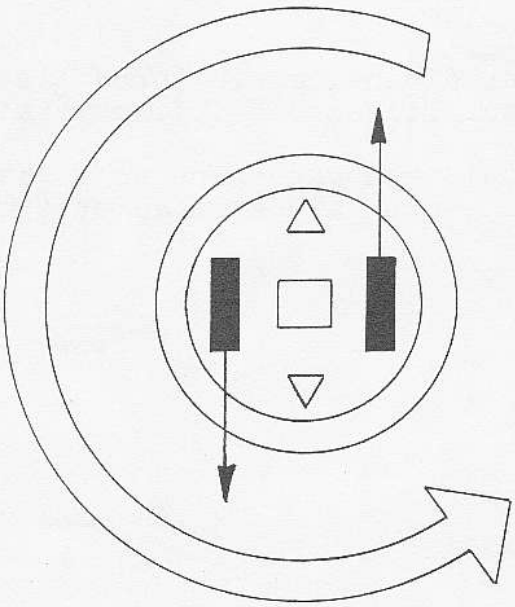


fig b.2.2.5

Se ora volete rispondere alle domande lasciate in sospeso nel paragrafo B.1, non dovete fare altro che premere i tasti corrispondenti...

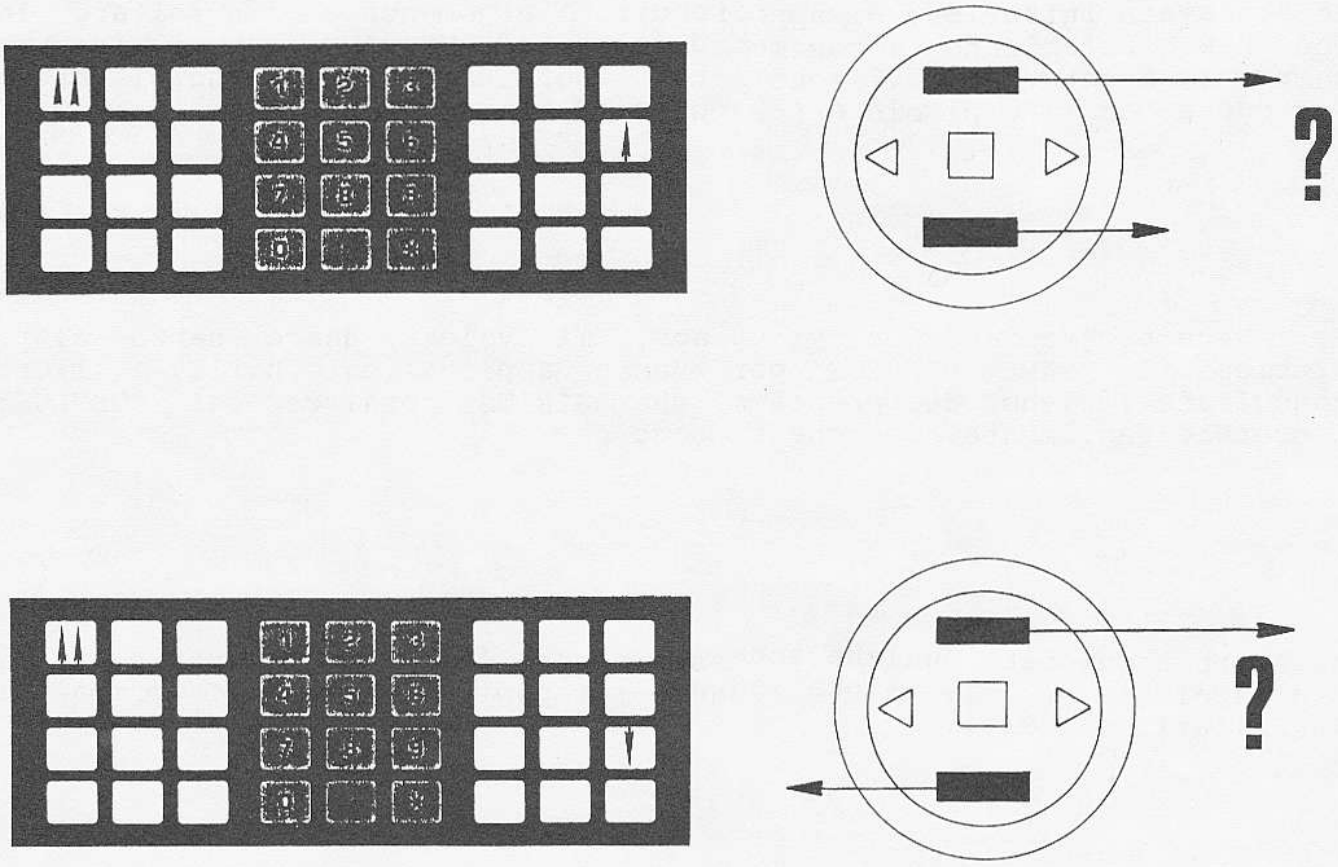


fig b.2.2.6

Vi consigliamo ora di provare tutti i movimenti che riuscite a produrre usando liberamente la tastiera della RF-CK: avete tutti gli strumenti per cavarvela nelle varie situazioni.

Provate a far seguire alla BASE MODULUS dei percorsi: per esempio "circumnavigare" la stanza senza urtare ostacoli, esplorare alcune zone, ... insomma tutto quello che vi viene in mente e puo´ servirvi a fare pratica con un controllo a due motori.

Questo sistema di controllo a "otto frecce" e´ semplice, ma permette comunque di raggiungere qualsiasi posizione voluta e con la BASE rivolta in qualsiasi direzione vogliate.

Se ora avete intenzione di approfondire l´argomento e controllare in modo piu´ sofisticato il movimento della BASE MODULUS con il vostro computer, queste prove vi sono servite per capire i principi base: procedete con il capitolo C (e poi D ed E).

Se invece programmare non vi piace, ma volete usare senza altri problemi la BASE MODULUS per tante applicazioni utili, allora consultate l´elenco dei programmi che SIRIUS fornisce per MODULUS: troverete facilmente cio´ che vi occorre.

Se avete acquistato qualche accessorio per la BASE (penna-plotter, aspirapolvere, ...), allora leggete il capitolo H od I prima di installarli e usarli.

B.3: Movimento della BASE con Commodore 64/128 e "filo diretto"

Questo paragrafo e´ riservato a chi possiede una BASE MODULUS con comando via cavo da Commodore 64 o 128.

Descriveremo qui un sistema per controllare il movimento della BASE con il vostro Commodore: questo sistema e´ analogo a quello usato con la tastiera dell´RF COMMAND KEYBOARD di cui simuleremo le funzioni.

Innanzi tutto introducete nel vostro Commodore questo programma BASIC e salvatelo su disco o cassetta.

```
10 REM PROGRAMMA DI MOVIMENTO PER BASE MODULUS
20 REM COPYRIGHT 1986 TECHNOLOGY TEAM S.R.L.
30 :
40 REM APERTURA DEL CANALE DI COMUNICAZIONE
50 OPEN5,2,0,CHR$(6)
60 GOSUB 500: REM   INIZIALIZZAZIONE DELLE COSTANTI
70 REM CICLO DI ATTESA COMANDI
80 GETA$
90 GOSUB 370
100 IF(( A$=B$ ) AND (( PS$<>N$ ) OR ( PD$<>N$ )))THEN 140
110 GOTO 80
120 REM FINE CICLO ATTESA COMANDI
130 :
140 PRINT"INOLTRO COMANDO"
150 GOSUB 300
160 REM INOLTRO COMANDO START PER TUTTI I MOTORI
170 PRINT#5,G$;
180 PRINT"COMANDO INOLTRATO"
190 GOSUB 300
200 REM CONTROLLO SE <SPAZIO> TUTTORA PREMUTO
210 IF PEEK(197)=60 THEN 190
220 PRINT"STOP"
230 REM INOLTRO COMANDO STOP PER TUTTI I MOTORI
240 PRINT#5,ST$;
250 PD$=N$:PS$=N$
260 GETA$
270 GOTO 80
280 :
290 REM SUBROUTINE INOLTRO COMANDI
300 PRINT#5,PD$;
310 GOSUB 700
320 PRINT#5,PS$;
330 GOSUB 700
340 RETURN
350 :
```

```
360 REM SUBROUTINE INTERPRETAZIONE TASTO PREMUTO
370 IFA$="1"THEN PD$=P5$:PRINT"AVANTI V SX":RETURN
380 IFA$="Q"THEN PD$=P6$:PRINT"AVANTI L SX":RETURN
390 IFA$="A"THEN PD$=P7$:PRINT"INDIETRO L SX":RETURN
400 IFA$="Z"THEN PD$=P8$:PRINT"INDIETRO V SX":RETURN
410 IFA$="3"THEN PS$=P1$:PRINT"AVANTI V DX":RETURN
420 IFA$="E"THEN PS$=P2$:PRINT"AVANTI L DX":RETURN
430 IFA$="D"THEN PS$=P3$:PRINT"INDIETRO L DX":RETURN
440 IFA$="C"THEN PS$=P4$:PRINT"INDIETRO V DX":RETURN
450 IFA$=CHR$(133) THEN PD$=AP$:PRINT"PENNA SU":RETURN
460 IFA$=CHR$(134) THEN PD$=CP$:PRINT"PENNA GIU'":RETURN
470 IFA$=CHR$(136) THEN PD$=RS$:PRINT"RICHIESTA STATO":RETURN
480 RETURN
490 :
500 REM SUBROUTINE DI INIZIALIZZAZIONE
510 N$="":B$=" "
520 PS$=N$:PD$=N$
530 G$=CHR$(0)+CHR$(83)+CHR$(83)+CHR$(255)
540 ST$=CHR$(0)+CHR$(67)+CHR$(67)+CHR$(255)
550 REM LETTURA POLINOMI MOVIMENTO DA TABELLA
560 P1$="":FORI=1TO10:READA:P1$=P1$+CHR$(A):NEXT
570 P2$="":FORI=1TO10:READA:P2$=P2$+CHR$(A):NEXT
580 P3$="":FORI=1TO10:READA:P3$=P3$+CHR$(A):NEXT
590 P4$="":FORI=1TO10:READA:P4$=P4$+CHR$(A):NEXT
600 P5$="":FORI=1TO10:READA:P5$=P5$+CHR$(A):NEXT
610 P6$="":FORI=1TO10:READA:P6$=P6$+CHR$(A):NEXT
620 P7$="":FORI=1TO10:READA:P7$=P7$+CHR$(A):NEXT
630 P8$="":FORI=1TO10:READA:P8$=P8$+CHR$(A):NEXT
640 AP$=CHR$(0)+CHR$(192)+CHR$(192)+CHR$(255)
650 CP$=CHR$(0)+CHR$(193)+CHR$(193)+CHR$(255)
660 RS$=CHR$(0)+CHR$(0)+CHR$(0)+CHR$(255)
670 RETURN
680 :
690 REM SUBROUTINE RITARDO TRA LE TRASMISSIONI
700 FOR RT = 1 TO 500 : NEXT : RETURN
710 :
720 DATA0,24,0,20,128,255,236,19,140,255:REM AV VEL SX
730 DATA0,24,0,10,128,255,246,9,146,255:REM AV LEN SX
740 DATA0,24,0,138,128,255,246,137,146,255:REM IND LEN SX
750 DATA0,24,0,148,128,255,236,147,140,255:REM IND VEL SX
760 DATA0,25,0,20,128,255,236,19,141,255:REM AV VEL DX
770 DATA0,25,0,10,128,255,246,9,147,255:REM AV LEN DX
780 DATA0,25,0,138,128,255,246,137,147,255:REM IND LEN DX
790 DATA0,25,0,148,128,255,236,147,141,255:REM IND VEL DX
800 REM FINE LISTATO
```

Alcuni tasti della tastiera del Commodore verranno usati per controllare la BASE MODULUS:

fig b.3.1

Passiamo in rassegna i tasti utilizzati.

F1 e F3 : riguardano il funzionamento della base con la penna-plotter o l´aspirapolvere: noi non li useremo per le prime prove.
Se possedete tali accessori e non resistete alla curiosita´, allora leggete il capitolo H o I.

F7 : serve a far "dimenticare" alla BASE un urto.

1 3 : motore (destro o sinistro) avanti veloce.

Q E : motore (destro o sinistro) avanti lento.

A D : motore (destro o sinistro) indietro lento.

Z C : motore (destro o sinistro) indietro veloce.

La colonna di tasti a sinistra controlla il motore di sinistra, quella a destra il motore di destra.

BARRA SPAZIO : fa eseguire il movimento che avete selezionato sulla tastiera.
Il movimento prosegue per tutto il tempo in cui tenete premuta la barra.
Quando volete cambiare tipo di movimento, rilasciate la barra, effettuate la selezione con i tasti visti sopra e ripremete la barra.

Mettete la BASE al centro della stanza, lontana da ostacoli e con il cavo libero di stendersi.

Tutto e´ pronto per partire: la BASE MODULUS e´ accesa e Commodore anche; il programma di controllo e´ caricato correttamente.
Se cosi´ non fosse accendeteli (non importa in che ordine) e provvedete a caricare il programma.
Se non vi ricordate come si prepara la BASE a funzionare, allora rileggete il paragrafo A.3 .

Premete il tasto E e poi Q : motore destro e sinistro avanti lento; premete la barra.

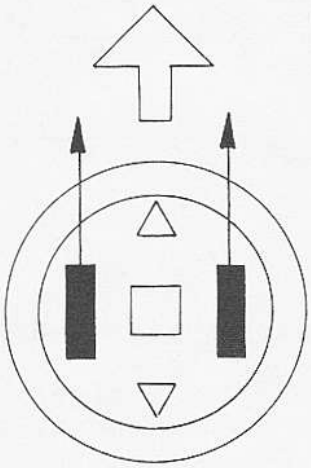


fig b.3.2

La BASE si muovera´ in avanti e continuera´ a muoversi fino a che tenete premuta la barra.

Provate piu´ volte.
Noterete che quando la BASE si muove si accendono un po´ di spie luminose.

Non ci vuole molto a capire il significato delle spie che si accendono e per ora ve lo lasciamo indovinare: con poche prove lo capirete.
Ma se proprio volete saperlo subito potete leggere il paragrafo C.2 .

Bene: ora premete di nuovo i tasti di avanti lento per i due motori e poi la barra ...

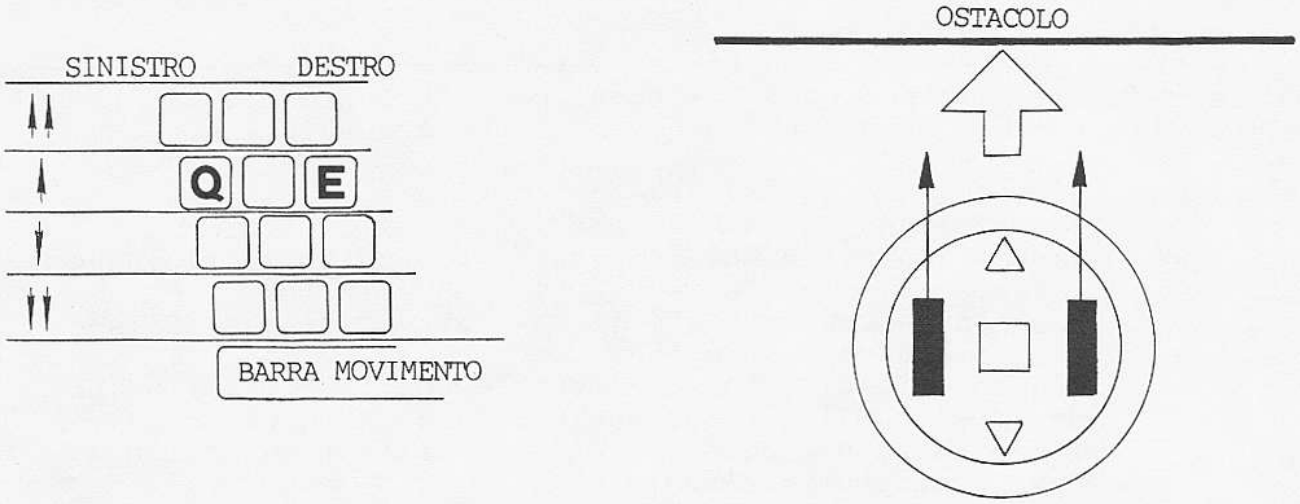


fig b.3.3

... e tenetela schiacciata fino a quando la BASE MODULUS non urti contro una parete o un altro ostacolo.

Che succede?

La BASE MODULUS e´ molto prudente e non vuole danneggiare ne´ se stessa ne´ gli altri.
Inoltre si sono accese delle spie rosse: la BASE "ricorda" l´urto subito.
Per dimenticarlo, la BASE MODULUS vuole proprio una vostra "autorizzazione ... scritta".
Premete a questo punto il tasto F7 seguito dalla barra: la BASE ha ora ricevuto la vostra autorizzazione.

Premete ora i tasti C e poi Z : indietro veloce destro e sinistro; premete la barra.

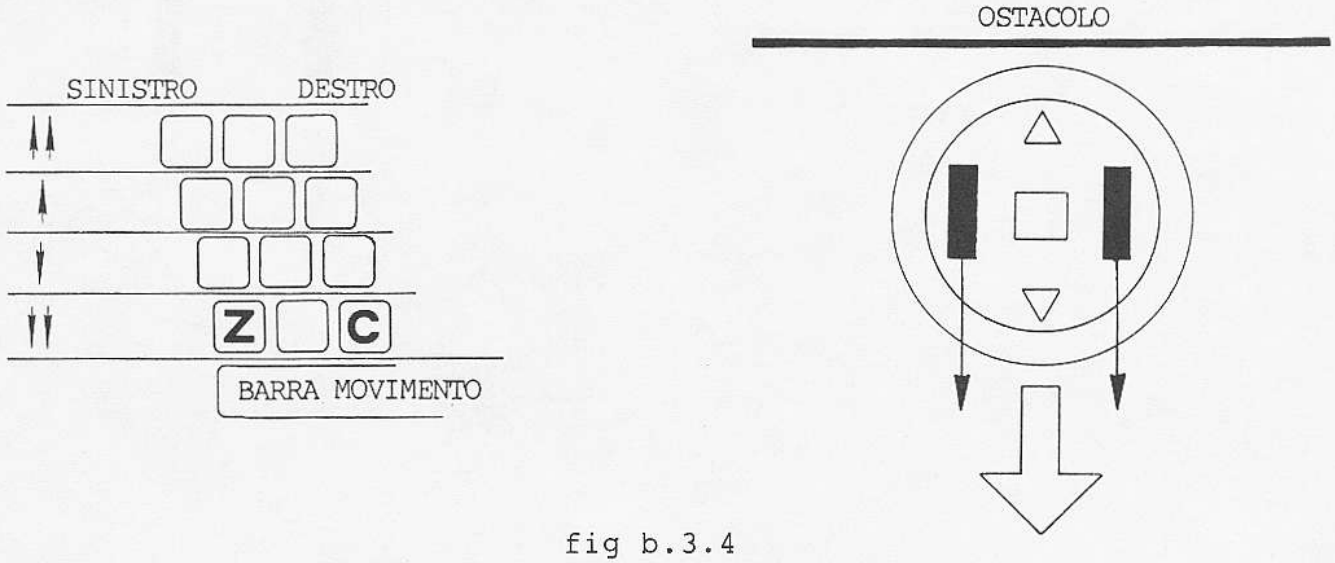


fig b.3.4

La base ritorna velocemente verso di voi: fermatela rilasciando la barra.

Se la BASE non reagisce come dovrebbe, provate a premere il tasto F7, la barra e poi procedete con i comandi voluti.

ATTENZIONE: la cosa piu´ noiosa che puo´ capitare con la BASE controllata via cavo e´ che il cavo si impigli in ostacoli o nella BASE stessa: soprattutto se si usano rotazioni.
Puo´ anche capitare che il cavo attorcigliato finisca con lo staccare le spine.
L´unico rimedio, volendo operare con il controllo via cavo, e´ quello di far muovere la BASE in un´area limitata, senza sorpassare i 2/3 della lunghezza del cavo "cordone ombelicale".

Se si stacca il cavo, riattaccatelo e riprendete il controllo di MODULUS premendo il tasto F7 e barra, riprendendo poi la sequenza interrotta. Se ci fossero ancora problemi, spegnete e riaccendete la BASE, fermate il programma su Commodore premendo RUN/STOP e poi fatelo ripartire con RUN.

Provate a far ruotare la BASE: premete il tasto avanti lento a destra (E) e poi il tasto indietro lento a sinistra (A): poi tenete premuta la barra.

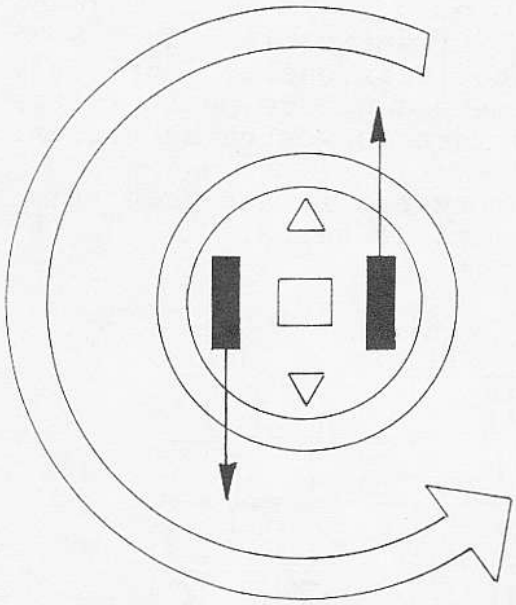


fig b.3.5

Se ora volete rispondere alle domande lasciate in sospeso nel paragrafo B.1, non dovete fare altro che premere i tasti corrispondenti e poi la barra ...

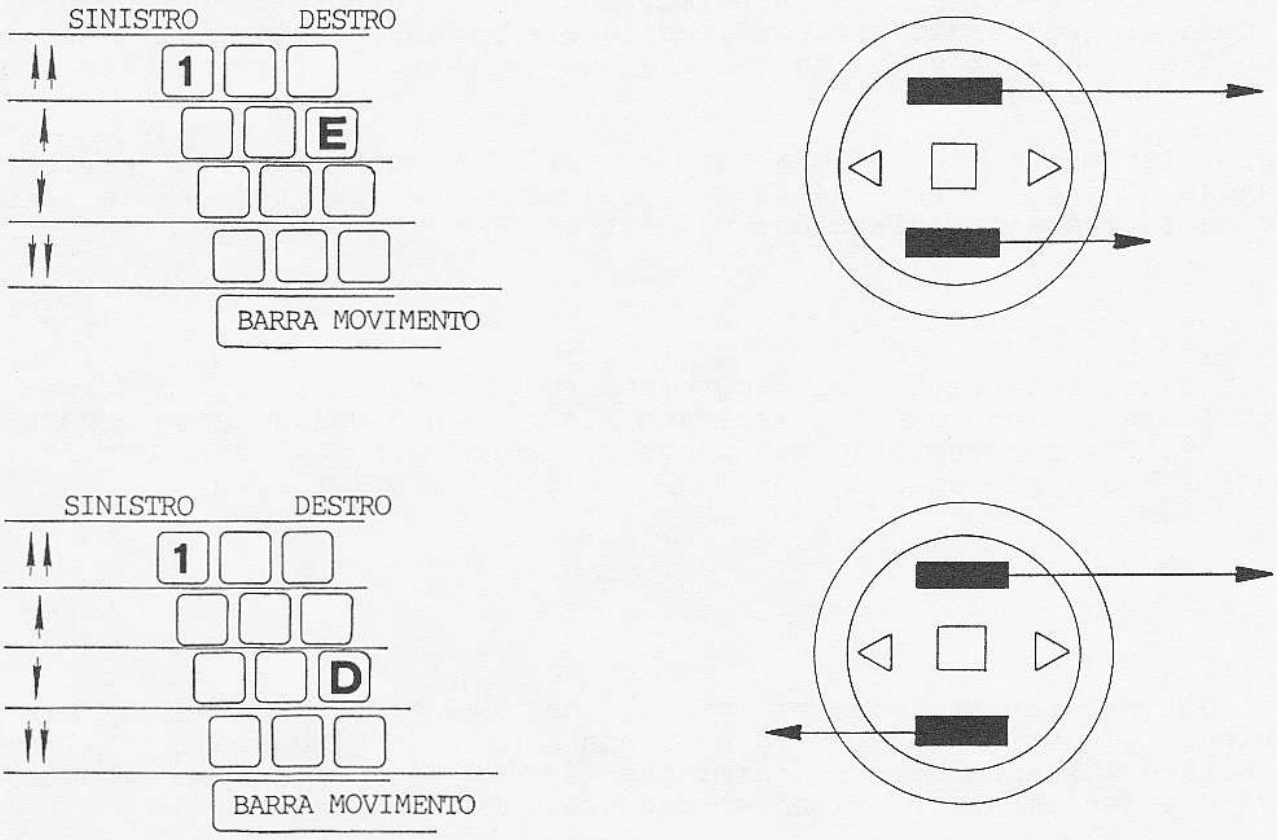


fig b.3.6

Vi consigliamo ora di provare tutti i movimenti che riuscite a produrre usando liberamente la tastiera : avete tutti gli strumenti per cavarvela nelle varie situazioni.

Provate a far seguire alla BASE MODULUS dei percorsi: per esempio "circumnavigare" degli ostacoli, esplorare alcune zone, ... insomma tutto quello che vi viene in mente e puo´ servirvi a fare pratica con un controllo a due motori.

Questo sistema di controllo a "dodici tasti" e´ semplice, ma permette comunque di raggiungere qualsiasi posizione voluta e con la BASE rivolta in qualsiasi direzione vogliate.

Se ora avete intenzione di approfondire l´argomento e imparare a controllare in modo piu´ sofisticato il movimento della BASE MODULUS con il vostro computer, queste prove vi sono servite per capire i principi base: proseguite con il capitolo C (e poi D ed E).

Se invece programmare non vi piace, ma volete usare senza altri problemi la BASE MODULUS per tante utili applicazioni, allora consultate l´elenco dei programmi che SIRIUS fornisce per MODULUS: troverete facilmente cio´ che vi occorre.

Se avete acquistato qualche accessorio per la BASE (penna-plotter, aspirapolvere, ...), allora leggete il capitolo H od I prima di installarli ed usarli.

C ---------- CONTROLLO DELLA BASE: LIVELLO AVANZATO -

C.1: Guida al movimento della BASE (livello 2)

Avete sperimentato alcune possibilita´ di movimento "teleguidando" MODULUS: tutto questo usando come intelligenza di guida il vostro cervello e come sensori i vostri occhi.
Se ora volete affidare la guida di MODULUS ad "un´intelligenza artificiale", ovvero ad un computer, che di intelligente ha ben poco, occorre studiare decisamente piu´ a fondo le possibilita´ di movimento della BASE.
Per i nostri scopi possiamo vedere la BASE come un "carrello" fatto da due ruote di diametro 81 millimetri dotate di moto indipendente, unite da un´asse di 271 millimetri di lunghezza.
Questo asse idealizza il fatto che le due ruote sono rigidamente vincolate fra loro dalla struttura della BASE.

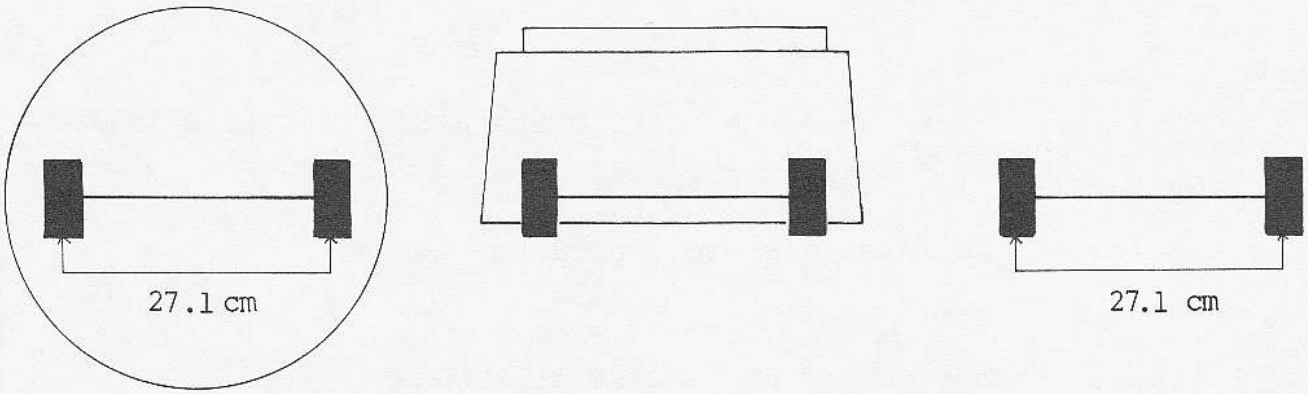


fig c.1.1

I due motori applicano, tramite le ruote, delle forze che "spingono" la BASE e la costringono a muoveersi.

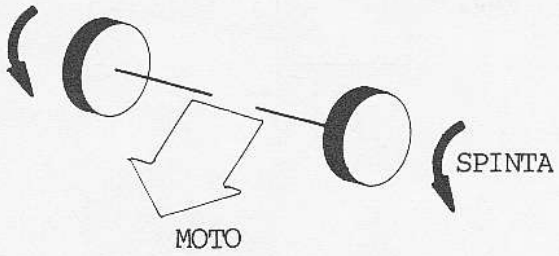


fig c.1.2

Dalla fisica sappiamo che l´effetto di una forza applicata ad un corpo in grado di muoversi, e´ quello di farlo accelerare.
In realta´ il nostro mondo reale e´ dominato dagli attriti: dopo un certo tempo , anche molto breve, il corpo sottoposto alla forza smettera´ di accelerare e viaggera´, per effetto di ogni sorta di attriti, ad una velocita´ costante, dipendente dalla forza applicata.

Questo e´ esattamente quello che succede quando, marce a parte, tenete il pedale dell´acceleratore della vostra automobile in una posizione fissa.

Quindi, semplificando per i nostri scopi, forza maggiore = velocita´ finale maggiore.
Rappresenteremo la velocita´ con delle frecce di lunghezza proporzionale: lunghezza della freccia 1 = velocita´ 1, lunghezza della freccia 2 = velocita´ 2, ...

Esaminiamo i movimenti prova i nel capitolo B.

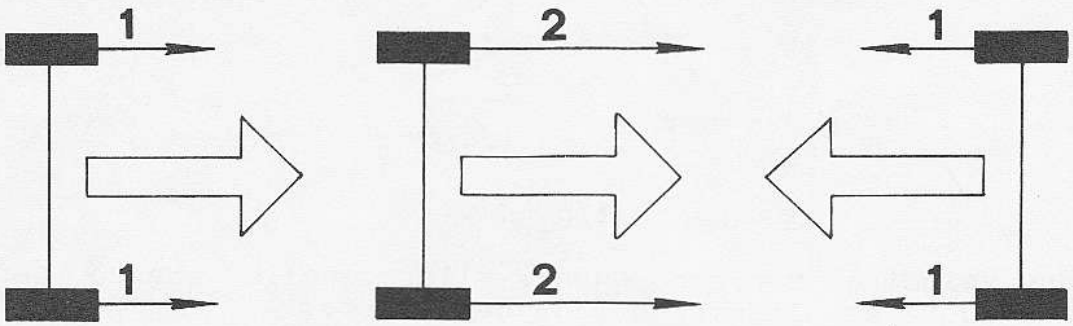


fig c.1.3

Se le velocita´ delle due ruote sono uguali e "applicate" nello stesso verso allora le de ruote copriranno, nello stesso tempo, la stessa distanza: abbiamo cioe´ un moto rettilineo.

Ricordiamo che l´asse disegnato serve solo a ricordare che le due ruote sono a distanza fissa: il loro movimento e´ invece completamente indipendente (cioe´ sono libere di muoversi a velocita´ diverse).

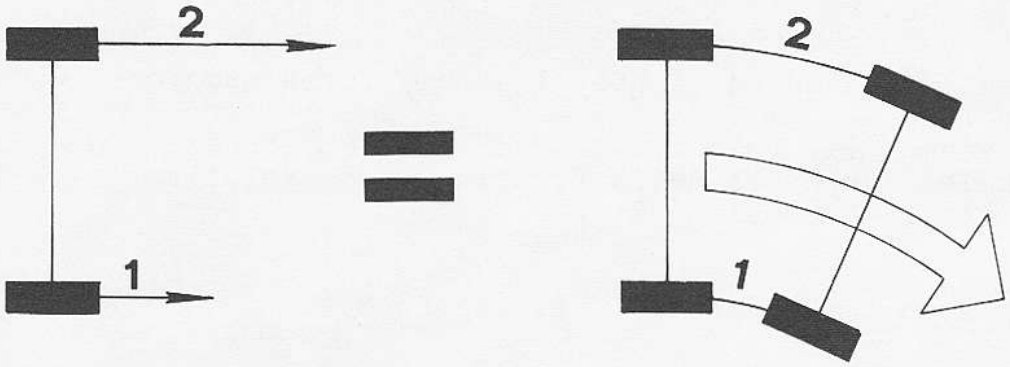
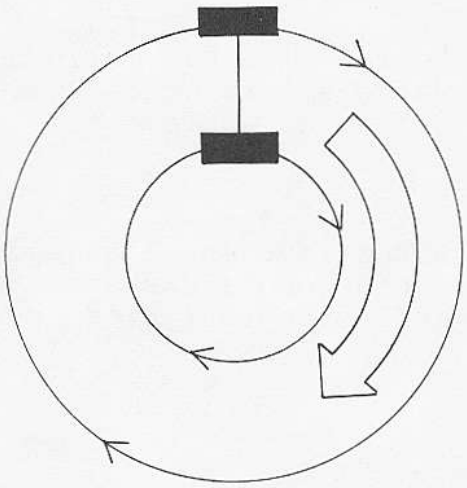


fig c.1.4

Se le due velocita´ non sono uguali allora, nello stesso tempo, una delle due ruote compie piu´ strada dell´altra.
Ma la distanza fra le due ruote non puo´ cambiare, perche´ l´asse le tiene insieme.
L´unica traiettoria con cui due ruote possono marciare a distanza fissa, ma a velocita´ diverse, e´ una curva.

La cosa diventa persino ovvia se pensiamo al nostro "carrello" che viaggia in cerchio attorno ad un punto (vedi fig c.1.5).



Le due ruote percorrono nello stesso tempo due circonferenze di lunghezza sicuramente differente: cioe´ viaggiano a velocita´ diverse.

fig c.1.5

Se, come negli esempi che abbiamo visto, la velocita´ delle due ruote e´ diversa, ma costante, allora la BASE percorrera´, se le diamo tempo (... e spazio!) a sufficienza, tutta la circonferenza.
Se le diamo meno tempo si limitera´ ad un pezzo, o arco, di circonferenza.
Il raggio di questa circonferenza si chiama "raggio di curvatura".
Il raggio si misura dal centro della circonferenza al centro della BASE, cioe´ alla meta´ dell´asse ideale che congiunge le due ruote.
Il centro della circonferenza viene detto centro di rotazione.

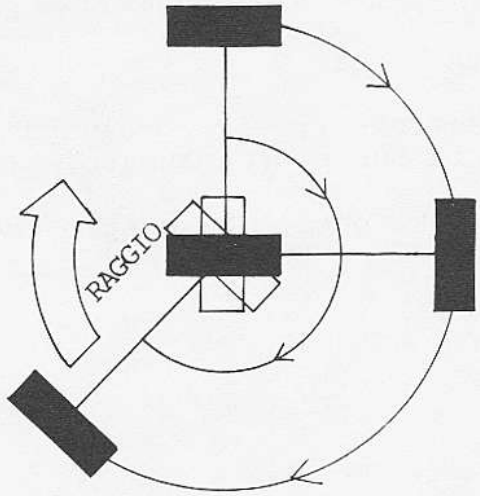


fig c.1.6

Sicuramente sia la velocita´ di percorrenza della curva che il suo "raggio" dipendono dalle velocita´ delle due ruote.

Infatti la cosa diventa ovvia se consideriamo il caso limite in cui una routa e´ a velocita´ zero, cioe´ "sta ferma", mentre l´altra si muove (vedi fig c.1.6).
In questa situazione il raggio della curva e´ uguale a mezzo asse, cioe´ 27,1 centimetri diviso 2: 13,55 cm.

ATTENZIONE: d´ora in poi, quando parleremo di traiettoria percorsa dalla BASE, inetnderemo quella percorsa dal suo centro, meta´ dell´asse ideale che congiunge le ruote, che prendiamo come punto di riferimento.

Per trovare il centro di rotazione (il punto attorno a cui si ruota) e, conseguentemente anche il raggio della curva, c´e´ un metodo grafico molto semplice. Vediamolo.

Prendere un foglio di carta a quadretti e scegliersi una unita´ di misura comoda, per esempio 1 quadretto = 1 centimetro.
Tracciare una linea di lunghezza corrispondente a quella del´asse fra le ruote (27,1 cm, cioe´ circa 27 cm).
Su ogni estremita´ della linea/asse tracciare, perpendicolarmete all´asse stesso, una freccia orientata nel verso del movimento provocato dalla corrispondente ruota.

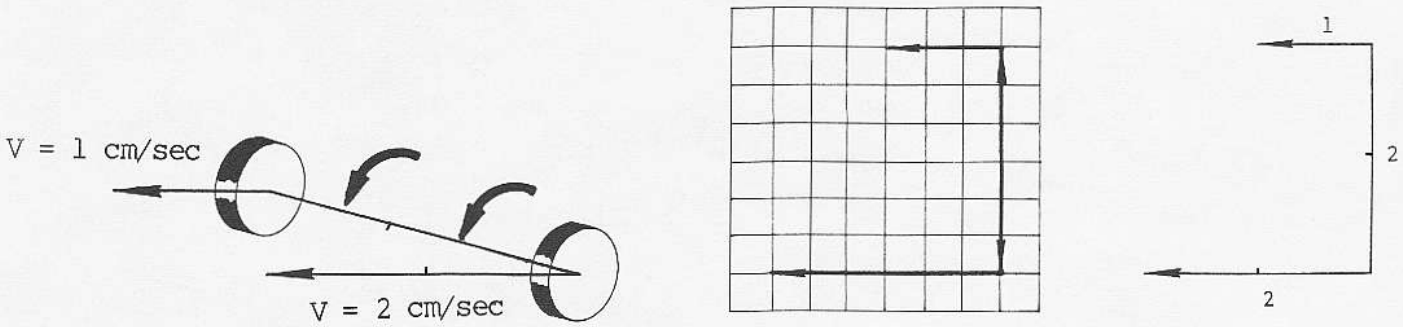


fig c.1.7

La lunghezza della freccia deve essere corrispondente alla velocita´ della corrispondente ruota.
Usare le stesse unita´ di lunghezza per tracciare asse e velocita´: se l´asse viene misurato in centimetri allora la velocita´ deve essere espressa in centimetri al secondo, se l´asse viene misurato in metri allora la velocita´ deve essere espressa in metri al secondo, ...

Poi unire le punte delle due frecce con una linea.

Il centro di rotazione si trova all´intersezione fra il prolungamento di questa linea con l´asse delle ruote od il suo prolungamento.

Il raggio della curva e´ la distanza di questo punto dal centro della BASE.

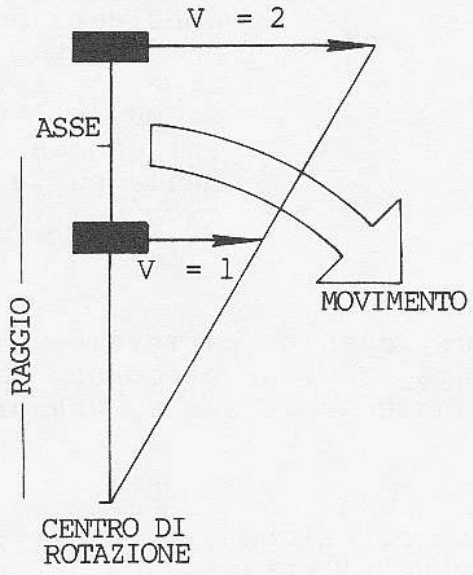


fig c.1.8

Se volete le cifre senza disegnare e misurare linee, allora potete usare delle formule.
Indichiamo con Vmag la velocita´ maggiore delle due ruote e con Vmin la minore, e usiamo i simboli aritmetici tipo BASIC (+,-,*,/).
Abbiamo:

RAGGIO = (ASSE/2) * (Vmag + Vmin) / (Vmag - Vmin)

FREQUENZA ROTAZIONE = (Vmag - Vmin) / (6,28 * ASSE)

ATTENZIONE: se la lunghezza dell´ASSE e´ espressa in cm allora il RAGGIO viene in centimetri.
Inoltre ASSE e le velocita´ vanno espresse in unita´ coerenti (se ASSE = X cm allora Vmag = Y cm/sec e Vmin = Z cm/sec): questa condizione e´ essenziale pre il calcolo della FREQUENZA DI ROTAZIONE.

Se i tempi sono espressi in secondi allora la FREQUENZA DI ROTAZIONE viene in giri/sec (se velocita´ = X cm/min allora frequenza = Y giri/min, ...).

Se volete farvi un´idea dei conti con delle cifre adatte alla BASE, ricordate che ASSE = 27,1 cm; anticipiamo che Vmag e Vmin possono variare da 1,36 a 6,6 cm/sec con la marcia bassa, o da 5,5 a 26,5 cm/sec con la marcia alta, come vedremo nel paragrafo C.2 .

Notate che nulla muta se le due ruote girano in senso inverso: basta tracciare le freccie nei due sensi e usare la stessa tecnica.

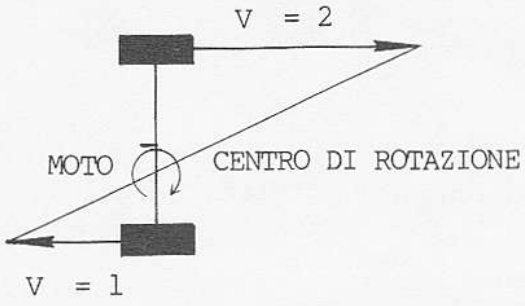


fig c.1.9

In questo caso la BASE, invece che ruotare attorno ad un punto esterno alla BASE stessa, ruota attorno ad un centro che sta in un punto del suo asse.

Nulla muta anche nelle formule per il calcolo del RAGGIO e della FREQUENZA DI ROTAZIONE, a patto di mettere il segno della velocita´: per la figura c.1.9, per esempio, Vmag = +2 e Vmin = -1.

Se poi le velocita´ delle due ruote che girano in senso opposto sono uguali, cioe´ Vmag = Vmin = (V=), abbiamo la seguente situazione.

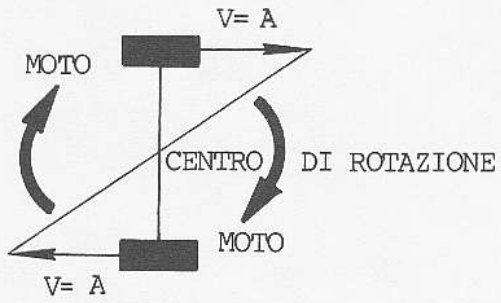


fig c.1.10

Il centro di rotazione coincide con il centro dell´asse della BASE, cioe´ la BASE ruota su se stessa.

In tal caso

RAGGIO = 0

e

FREQUENZA ROTAZIONE = 2 * (V=) / (6,28 * ASSE)

che si ottengono ponendo nelle formule precedenti Vmin = - Vmag.

Ma torniamo al nostro problema delle curve ...

PRIMA AGGIUNTA:

Confrontiamo questi due percorsi:

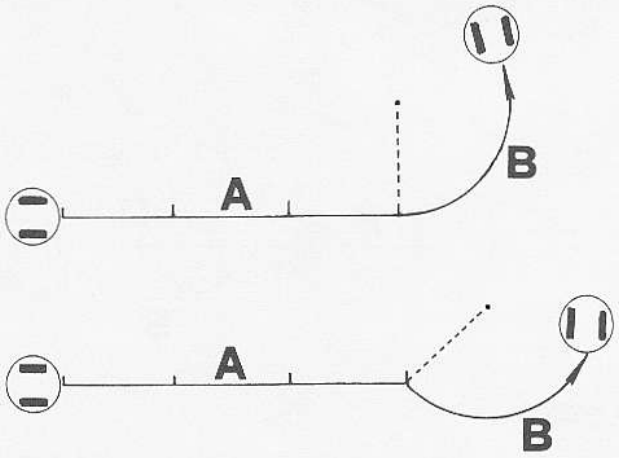


fig c.1.11

In tutti e due i casi abbiamo un tratto rettilineo di tre unita´ seguito da un arco "sinistro" di circonferenza con raggio di una unita´...ma che differenza.
Manca allora un parametro nella nostra descrizione: vediamo un po´...
Nel caso 1 c´e´ continuita´ fra il tratto rettilineo A e la curva B.
Nel caso 2 no: fra la parte A e la B c´e´ uno spigolo.

Questo significa che, nel caso 2, nel passare dalla parte A alla parte B c´e´, nascosta, una "rotazione attorno al proprio centro" che permette di fare lo spigolo!

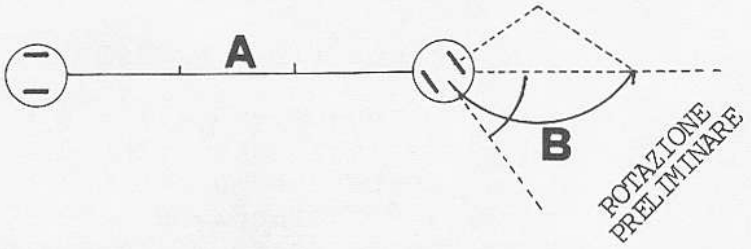


fig c.1.12

Ora o si procede per tentativi o si devono fare due conti: facciamoli.
Tiriamo una linea, che chiameremo C, fra il centro di rotazione ed il punto di inizio della curva B e prolunghiamo la parte rettilinea A.

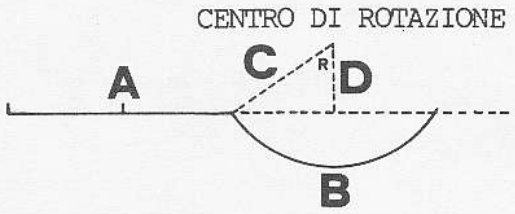


fig c.1.13

Chiamiamo D la distanza fra il centro di rotazione e il prolungamento di A.
Per motivi geometrici l´angolo di cui deve ruotare su se stessa la BASE e´ uguale all´angolo acuto R, sotteso dalle due linee D e C.
Conoscendo D, per esempio D = 0,6 unita´, con un po´ di trigonometria si trova l´angolo di rotazione R:

$$\cos R = D / RAGGIO$$

ovvero

$$R = \arccos (D / RAGGIO)$$

cioe´ nel nostro esempio: cos R = 0,6/1 = 0,6
ovvero: R = arcos 0,6 ; se valutate arcos 0,6 (per esempio con il vostro calcolatore) otterrete R = 0,927 radianti o, tenuto conto che gradi = 57,32 * radianti, R = 53,13 gradi.
In totale, nel nostro esempio/caso 2 , occorre fra i movimenti A e B una rotazione oraria (destra) di 53 gradi.

SECONDA AGGIUNTA:

Fino ad ora abbiamo lavorato a velocita´ costante: questo ci assicura curve con raggi di curvatura costante.
Cioe´ ogni curva e´ sicuramente un arco di una circonferenza di opportuno raggio e centro.
Piu´ in generale, per avere un raggio di curvatura costante, basta che sia costante il rapporto fra le velocita´ delle due ruote.

MODULUS puo´ anche essere controllato in accelerazione: semplicemente finora abbiamo imposto accelerazione zero, cioe´ velocita´ costante.
Se introduciamo una accelerazione diversa da zero nelle curve, possiamo guidare la BASE su traiettorie a curvatura non costante: per esempio curve che si stringono o si allargano (= raggio di curvatura che diminuisce o aumenta).
Il compito di studiare queste traiettorie lo lasciamo ai piu´ intraprendenti sperimetatori o ai piu´ curiosi in fisica.
Noi decidiamo, per semplicita´, di continuare ad usare velocita´ costanti, cioe´ accelerazione nulla.

Arrivati a questo punto siamo in grado di ricavare i parametri di moto di ognuna delle due ruote della BASE per un problema di questo tipo:

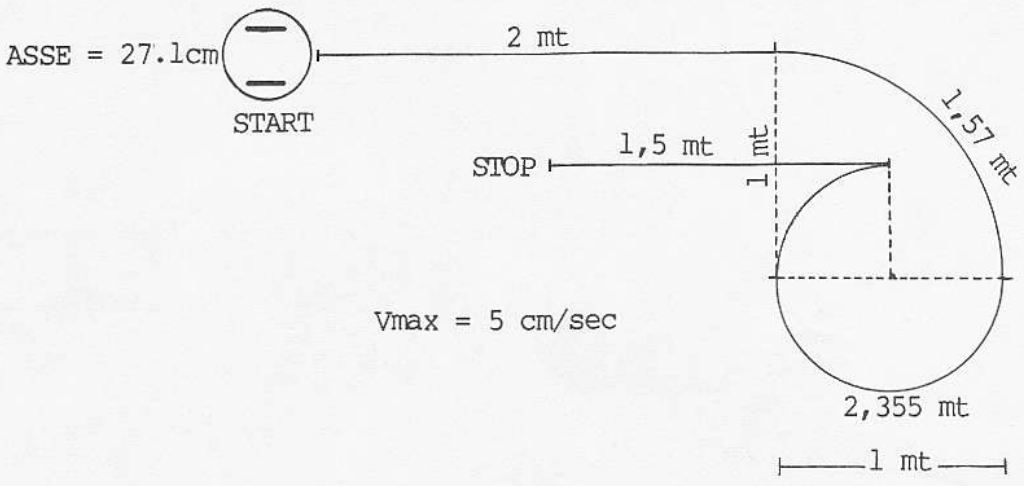


fig c.1.14

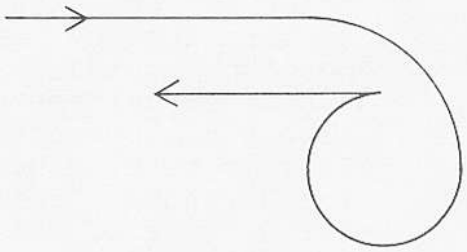
Provate a cavarvela da voi ... e´ molto istruttivo e si scoprono tanti particolari interessanti.

Ma non c´e´ un sistema piu´ spiccio?

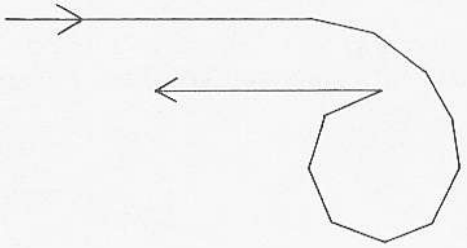
Beh, se non vi interessano i movimenti sinuosi ed eleganti ... si puo´ fare!

Ogni traiettoria puo´ essere approssimata a piacere usando solo spezzate, cioe´ linee dritte e spigoli.
Per noi questo vuol dire che ogni percorso puo´ essere eseguito con vari gradi di precisione usando solo traslazioni (= movimenti in linea retta) e rotazioni attorno al proprio asse.

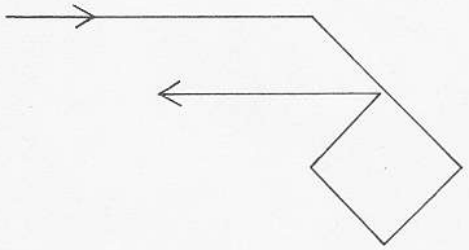
Il nostro percorso curvilineo ...



...puo´ diventare cosi´...



...o cosi´...



...o se vi interessa solo raggiungere la posizione finale!

fig c.1.15

Il movimento e' un po' a scatti e piu' "robotico".

In compenso e' piu' facile calcolare solo distanze e angoli di rotazione, piuttosto che centri e raggi di curvatura... provateci!

Vogliamo vedere?
Prendiamo il percorso "di prova" cosi' approssimato:

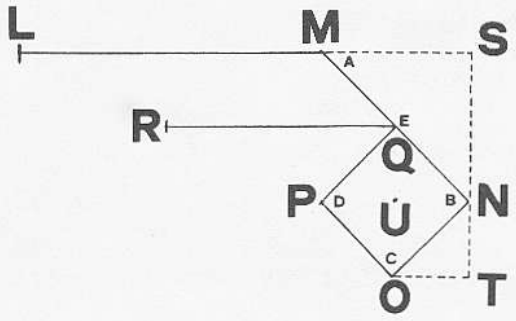


fig c.1.16

Ci serve solo un po' di geometria:

- l'angolo "a" e' l'angolo acuto di un triangolo isoscele e rettangolo: deve essere per forza di 45 gradi.
- gli angoli "b","c" e "d" sono retti: 90 gradi.
- l'angolo "e" e' il supplementare di un angolo acuto di triangolo isoscele e rettangolo (45 gradi): deve essere 360 - 45 = 315 gradi.

... e un pizzico di Pitagora:

Indichiamo i segmenti con i nomi dei due estremi; ad esempio la parte rettilinea e' il segmento LM.
Indichiamo X elevato al quadrato con (X)^2

Conosciamo gia' che LM = 200 cm .
Sappiamo allora che PM = PN = MS = SN = 100 cm
Inoltre UP = UO = UN = UQ = OT = TN = 50 cm .

- (SM)^2 + (SN)^2 = (MN)^2 ; ma SM = SN perche' il triangolo MSN e' isoscele e quindi vale 2 * (SM)^2 = (MN)^2 cioe' 2 * (100)^2 = (MN)^2 ovvero (MN)^2 = 20000.
  In totale MN = 141,4 cm .
- (ON)^2 = (NT)^2 + (TO)^2 ; ma il triangolo NTO e' isoscele per cui NT = TO e dunque vale (ON)^2 = 2 * (TO)^2 cioe' (ON)^2 = 5000.
  In totale ON = OP = PQ = 70,7 cm .

I piu' furbi avrebbero notato subito che ON doveva essere la meta' di MN e avrebbero evitato il secondo conto.

E ora?

Per i pezzi rettilinei nessun problema: i due motori devono andare nello stesso verso e con la stessa velocita´.

Dobbiamo solo calcolare il tempo di percorrenza, per sapere per quanto tempo dovremo tenere i due motori alla velocita´ stabilita.
Prendiamo velocita´ V = 5 cm/sec, la massima prescritta dal problema.
Usiamo TEMPO = SPAZIO / VELOCITA´ .

t-LM = 200 / 5 = 40 secondi .

t-MN = 141,4 / 5 = 28,28 secondi .

t-NO = t-OP = t-PQ = 70,7 / 5 = 14,14 sec .

t-QR = 150 / 5 = 30 sec .

Poi le rotazioni: qui i due motori vanno alla stessa velocita´ ma in sensi opposti.

Dobbiamo calcolare la "velocita´" di rotazione; scegliamo per i motori (V=) = 5 cm/sec e usiamo la:

FREQUENZA ROTAZIONE = 2 * (V=) / (6,28 * ASSE)

cioe´ per noi: 2 * 5 / (6,28 * 27,1) = 0,0587 giri/sec

ora TEMPO ROTAZIONE = NUMERO GIRI / FREQUENZA ROTAZIONE in giri/sec

o TEMPO ROTAZIONE = GRADI / 360 * FREQUENZA ROTAZIONE in giri/sec

Calcoliamo i tempi di rotazione, necessari per ottenere gli angoli voluti:

t-45 = 45 / 0,0587 * 360 = 2,13 sec .

t-90 = 2 * t-45 = 4,26 sec .

t-315 = 7 * t-45 = 14,9 sec .

Riassumiamo i parametri di moto (destro = dx, sinistro = sx):

TRATTO 1 (LM) : mot. dx avanti a V = +5 cm/sec per t-LM = 40 sec,
mot. sx avanti a V = +5 cm/sec per t-LM = 40 sec .

TRATTO 2 (a) : mot. dx indietro a V = -5 cm/sec per t-45 = 2,13 sec,
mot. sx avanti a V = +5 cm/sec per t-45 = 2,13 sec .

TRATTO 3 (MN) : mot. dx avanti a V = +5 cm/sec per t-MN = 28,28 sec,
mot. sx avanti a V = +5 cm/sec per t-MN = 28,28 sec .

TRATTO 4 (b) : mot. dx indietro a V = -5 cm/sec per t-90 = 4,26 sec,
mot. sx avanti a V = +5 cm/sec per t-90 = 4,26 sec .

TRATTO 5 (NO) : mot. dx avanti a V = +5 cm/sec per t-NO = 14,14 sec,
mot. sx avanti a V = +5 cm/sec per t-NO = 14,14 sec .

TRATTO 6 (c) : mot. dx indietro a V = -5 cm/sec per t-90 = 4,26 sec,
mot. sx avanti a V = +5 cm/sec per t-90 = 4,26 sec .

TRATTO 7 (OP) : mot. dx avanti a V = +5 cm/sec per t-OP = 14,14 sec,
mot. sx avanti a V = +5 cm/sec per t-OP = 14,14 sec .

TRATTO 8 (d) : mot. dx indietro a V = -5 cm/sec per t-90 = 4,26 sec,
mot. sx avanti a V = +5 cm/sec per t-90 = 4,26 sec .

TRATTO 9 (PQ) : mot. dx avanti a V = +5 cm/sec per t-PQ = 14,14 sec,
mot. sx avanti a V = +5 cm/sec per t-PQ = 14,14 sec .

TRATTO 10 (e) : mot. dx avanti a V = +5 cm/sec per t-315 = 14,9 sec,
mot. sx indietro a V = -5 cm/sec per t-315 =14,9 sec.

TRATTO 11 (QR): mot. dx avanti a V = +5 cm/sec per t-QR = 30 sec,
mot. sx avanti a V = +5 cm/sec per t-QR = 30 sec .

Risolviamo, per confronto e per consolare quelli che lo avevano diligentemente fatto, il problema originale con moto "sinuoso", cioe´ senza spigoli.

Scomponiamo il problema nei vari pezzi:

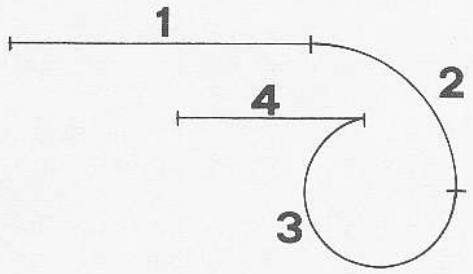


fig c.1.17

Esaminiamo ora i singoli tratti.

TRATTO 1

Qui e´ facile: si tratta di andare avanti dritti.
Viaggiamo a velocita´ massima prescritta dal probelma: 5 cm/sec .

Calcoliamo il tempo per cui occorre tenere questa velocita´ per coprire la distanza richiesta: TEMPO = SPAZIO / VELOCITA´

Per noi t-1 = 200 / 5 = 40 sec .

TRATTO 2

Cioe´ 90 gradi di curva a destra con raggio 100 cm.
Il centro di rotazione e´ esterno alla BASE e a destra.
Dunque abbiamo motori tutti e due avanti ma a velocita´ diverse.
E´ anche evidente che V destro < V sinistro.
Usiamo le formule:

RAGGIO = (ASSE/2) * (Vmag + Vmin) / (Vmag - Vmin)

FREQUENZA ROTAZIONE = (Vmag - Vmin) / (6,28 * ASSE)

Perche´ il problema abbia un´unica soluzione occorre fissare sia il raggio che la frequenza, oppure il raggio e una delle due velocita´.

Fissiamo allora V sinistra = Vmag = +5 cm/sec e facciamo i conti.

Ricaviamo Vmin dalla formula del RAGGIO:

Vmin = Vmag * ((2*RAGGIO) - ASSE) / ((2*RAGGIO) + ASSE)

cioe´ 5 * ((200 - 27,1)/(200 + 27,1)) = 3,8 cm/sec circa

dunque Vmin = +3,8 cm/sec : ora calcoliamo la FREQUENZA DI ROTAZIONE

FREQUENZA ROTAZIONE = (5 - 3,8)/(6,28 * 27,1) = 0,007 giri/sec circa

90 gradi sono 1/4 di giro; calcoliamo il tempo di percorrenza dell´arco;

ora TEMPO ROTAZIONE = NUMERO GIRI / FREQUENZA ROTAZIONE in giri/sec

cioe´ t-2 = 0,25 / 0,007 = 35,71 sec

TRATTO 3

Cioe´ 270 gradi di curva a destra con raggio 50 cm.
Il centro di rotazione e´ esterno alla BASE e a destra.
Dunque abbiamo motori tutti e due avanti ma a velocita´ diverse.
E´ anche evidente che V destro < V sinistro.
Come prima fissiamo allora V sinistra = Vmag = +5 cm/sec e facciamo i conti usando la formula usata sopra:

Vmin = Vmag * ((2*RAGGIO) - ASSE) / ((2*RAGGIO) + ASSE)

cioe´ 5 * ((100 - 27,1)/(100 + 27,1)) = 2,86 cm/sec circa

dunque Vmin = +2,86 cm/sec : ora calcoliamo la FREQUENZA DI ROTAZIONE

FREQUENZA ROTAZIONE = (5 - 2,86)/(6,28 * 27,1) = 0,0125 giri/sec

270 gradi sono 3/4 di giro; calcoliamo il tempo di percorrenza dell´arco;

ora TEMPO ROTAZIONE = NUMERO GIRI / FREQUENZA ROTAZIONE in giri/sec

cioe´ t-2 = 0,75 / 0,0125 = 60 sec

TRATTO 4

Qui la base e´ rivolta verso destra e deve tornare indietro: se vogliamo che arrivi con il "muso" a sinistra in fondo al percorso, allora occorre fare prima una rotazione di 180 gradi a destra o a sinistra.
I motori devono andare controversi e alla stessa velocita´.
Scegliamo la massima velocita´ prevista dal problema ((V=) = 5 cm/sec) e facciamo una rotazione oraria (destra).

V destra = -5 cm/sec e V sinistra = +5 cm/sec

Calcoliamo la frequenza di rotazione conseguente:

FREQUENZA ROTAZIONE = 2 * (V=) / (6,28 * ASSE)

cioe´ per noi: 2 * 5 / (6,28 * 27,1) = 0,0587 giri/sec

ora TEMPO ROTAZIONE = NUMERO GIRI / FREQUENZA ROTAZIONE in giri/sec

o TEMPO ROTAZIONE = GRADI / 360 * FREQUENZA ROTAZIONE in giri/sec

Calcoliamo il tempo di rotazione, necessario per ottenere l´angolo voluto:

t-4-1 = 0,5 / 0,0587 = 8,51 sec

Ora si tratta di andare avanti dritti per 150 cm .
Viaggiamo a velocita´ massima prescritta dal probelma: 5 cm/sec .

Calcoliamo il tempo per cui occorre tenere questa velocita´ per coprire la distanza richiesta: TEMPO = SPAZIO / VELOCITA´

Per noi t-4-2 = 150 / 5 = 30 sec .

Riassumiamo i parametri di moto (destro = dx, sinistro = sx):

TRATTO 1 : mot. dx avanti a V = +5 cm/sec per t-1 = 40 sec,
mot. sx avanti a V = +5 cm/sec per t-1 = 40 sec .

TRATTO 2 : mot. dx avanti a V = +3,8 cm/sec per t-2 = 35,71 sec,
mot. sx avanti a V = +5 cm/sec per t-2 = 35,71 sec .

TRATTO 3 : mot. dx avanti a V = +2,86 cm/sec per t-3 = 60 sec,
mot. sx avanti a V = +5 cm/sec per t-3 = 60 sec .

TRATTO 4-1 : mot. dx indietro a V = -5 cm/sec per t-4-1 = 8,51 sec,
mot. sx avanti a V = +5 cm/sec per t-4-1 = 8,51 sec .

TRATTO 4-2 : mot. dx avanti a V = +5 cm/sec per t-4-2 = 30 sec,
mot. sx avanti a V = +5 cm/sec per t-4-2 = 30 sec .

Riassumiamo la situazione.

Il metodo delle curve e´ piu´ complicato ma produce tutte le traiettorie volute (a curvatura costante o anche no, se si usa accelerazione non nulla) .
La maggiore complicazione risulta evidente se provate a tracciare una traiettoria qualsiasi a mano (non scelta e disegnata con riga e compasso, come l´esempio) e poi provate a programmarla con questo sistema.

Il sistema delle spezzate e delle rotazioni e´ meno elegante ma il suo uso risente molto meno della complessita´ della traiettoria da seguire approssimatamente.
Non e´ mai troppo difficile ridurre ad una spezzata la traiettoria che si desidera.

Anticipiamo che pero´ se si usa l´accessorio penna-plotter per disegnare con la BASE la scelta e´ obbligata: occorre usare il metodo delle curve adattato alla posizione della penna.
Vedremo nel paragrafo H.2.3 il perche´.

C.2: Indicatori luminosi e sensori di urto

La BASE MODULUS porta sulla parte superiore una serie di indicatori luminosi che danno un´idea immediata delle "attivita´ in corso".
Se avete provato la BASE, come indicato nel capitolo B, dovreste aver gia´ capito il loro significato, che e´ molto intuitivo.

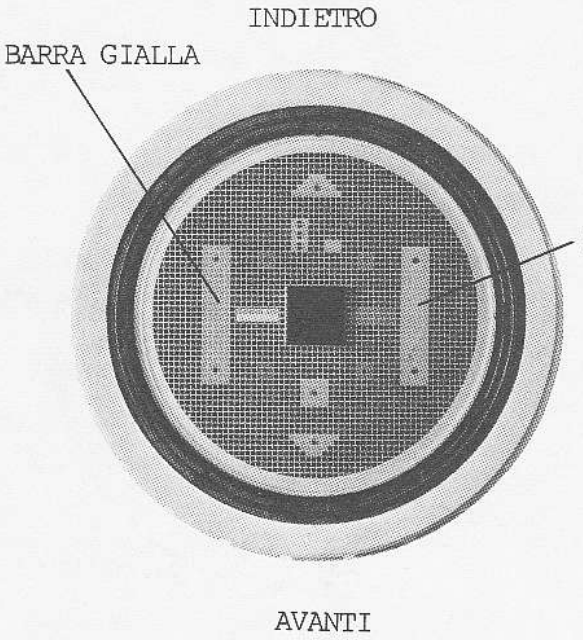


fig c.2.1

Guardiamo la BASE dall´alto.

- Le due strisce gialle simboleggiano le ruote/gruppo motore.
  Ogni motore ha due spie.

  Quando il motore viene pilotato, la spia corrispondente alla direzione del moto di quella ruota lampeggia.
  La frequenza di lampeggio e´ proporzionale alla velocita´ del moto.
  Abbiamo cosi´ un sistema estremamente rapido per renderci conto se abbiamo pilotato correttamente i motori.

- Ci sono poi due triangoli gialli.
  I triangoli puntano verso le due direzioni di possibile moto traslatorio: avanti e dietro.
  La direzione avanti e´ rappresentata dal triangolo giallo dalla parte della spia di accensione, quella indietro dal triangolo sopra il simbolo di batteria.
  Queste spie si accendono quando entrambe le ruote girano nello stesso senso, assicurando una ben definita direzione di marcia.
  Per esempio, se entrambe i motori-ruote marciano in avanti, anche a velocita´ diverse, allora si accendera´ la spia gialla di "direzione avanti".

- Alla sinistra del simbolo di batteria ci sono tre spie di stato batteria:

  VERDE = batterie con carica massima

  GIALLO = batterie con carica media

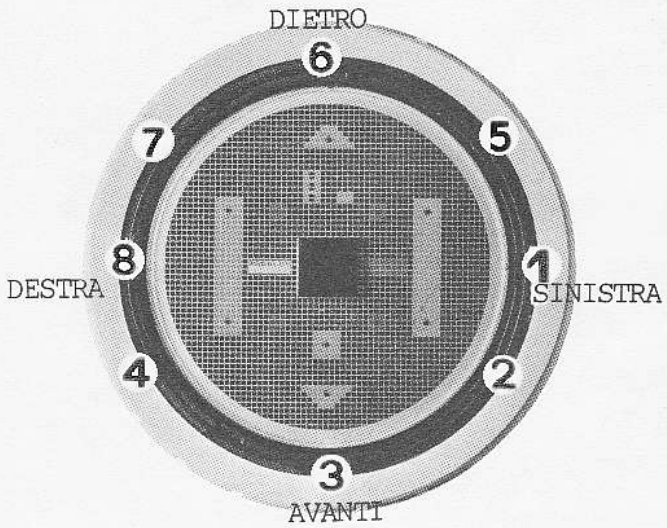
  ROSSO = batterie scariche

Quando la spia rossa si accende occorre tempestivamente provvedere alla ricarica della batterie: se non lo si fa si rischia di non riuscire piu´ a ricaricarle.
Consultate il paragrafo A.3 .

- Quattro quadrati rossi contengono delle spie rosse, che si accendono quando qualcuno dei sensori della BASE MODULUS si "accorge" di un urto.
  Quando si accendono le spie, cioe´ ad urto avvenuto, i motori della BASE si fermano automaticamente, per sicurezza.

Vediamo bene il sistema di sensori di urto di cui e´ dotato MODULUS.

Ci sono 8 sensori di urto dislocati ogni 45 gradi lungo la circonferenza della BASE.



I sensori sono numerati come in figura c.2.2 .
Colloquiando con MODULUS potrete sapere quale degli otto sensori ha recepito un urto.

fig c.2.2

Si leggera´ un byte in cui ognuno degli otto bit che lo compongono riporta lo stato del relativo sensore di urto:

Il corrispondente bit e´ a 1 quando il sensore (bumper) ha recepito l´urto.

Per esempio, se la BASE ha urtato in una zona compresa fra i sensori 2 e 3 (cioe´ avanti ma leggermete a sinistra), il byte di stato BUMPERS conterra´:

BYTE BUMPERS	0	0	0	0	0	1	1	0	= 00000110 bin = 06 hex = 6 dec
	msbit							lsbit	

Cioe´ sia il sensore 2 che il 3 hanno registrato l´urto.

I bumpers sono costituiti da delle levette di plastica con un lato appoggiato alla gonnellina mobile della BASE.
Se schiacciate, vanno a premere un interruttore a membrana che si chiude, rivelando l´urto.
Le levette si possono togliere e rimontare agevolmente infilandole nelle apposite cave nella parte inferiore della BASE, sotto la gonnellina.
Le cave riservate alle levette dei bumpers si individuano perche´ sono in corrispondenza degli interruttori a membrana (piazzuole ovali metalliche).

Quando avviene un urto, dicevamo, la BASE ferma i motori e accende le quattro spie rosse di urto.
Per ripartire occorrera´ inviare nuovi comandi di movimento alla BASE.

Prima di ripartire sara´ comunque opportuno richiedere lo stato della BASE: questo, come vedremo meglio nel paragrafo D.5, e´ un messaggio che arriva dalla BASE e che contiene tutte le informazioni sul suo stato attuale di funzionamento.

Il byte di BUMPERS e´ contenuto in questo messaggio.
Leggendo il byte BUMPERS capiremo in quale direzione c´e´ stato urto e quindi in che direzione ci conviene far ripartire la BASE.

Inoltre, richiedendo lo stato BASE si causera´ lo spegnimento delle spie di urto.

Quello che abbiamo fatto nel capitolo B, premendo il tasto SR della RF-CK o il tasto F7 del Commodore, in caso di connessione via filo, e´ stato proprio di richiedere alla BASE il messaggio di stato .

C.3 Guida ai motori

Prima di passare a programmare il controllo dei motori della BASE MODULUS diamo un rapido sguardo ai motori e ai principi del loro controllo.

La BASE e´ fornita di due motori in corrente continua con tensione massima di pilotaggio di 24 volt .
La loro massima velocita´ senza carico e´ di 5500 giri/min all´albero motore.

Ma sotto l´azione del peso della BASE, degli ingranaggi e degli attriti, e del controllo elettronico la massima velocita´ operativa scende in modo non definibile con esattezza.
Per fissare le idee ddiciamo che la massima velocita´ raggiungibile e´ di circa 4000 giri/min .

Ricordiamo che fra l´asse motore e l´asse delle ruote abbiamo una serie di ingranaggi di demoltiplica che realizzano il "cambio" visto nel paragrafo B.1 .

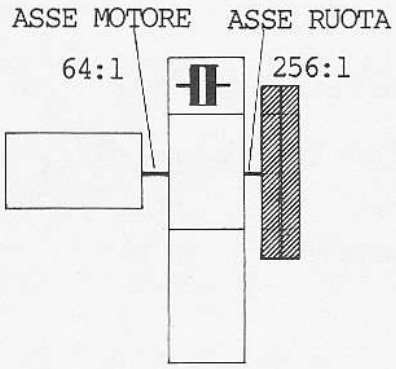


fig c.3.1

Il rapporto totale di demoltiplica fra asse motore e asse ruote e´ di 256:1 o di 64:1 a secondo che si selezioni la marcia bassa o quella alta.
La frequenza di rotazione sull´asse ruote, corrispondente a 4000 giri/min sull´asse motore, e´ rispettivamente di 15,6 e di 62,5 giri/min .

Considerando che il diametro della ruota e´ di 81 mm, ricaviamo che la massima velocita´ lineare raggiungibiile dalla BASE e´ rispettivamente di 3,96 e di 15,9 metri/minuto (cioe´ 6,6 e 26,5 cm/sec).

Dopo questa carrellata vediamo il sofisticato sistema microelettronico di controllo dei motori di MODULUS.

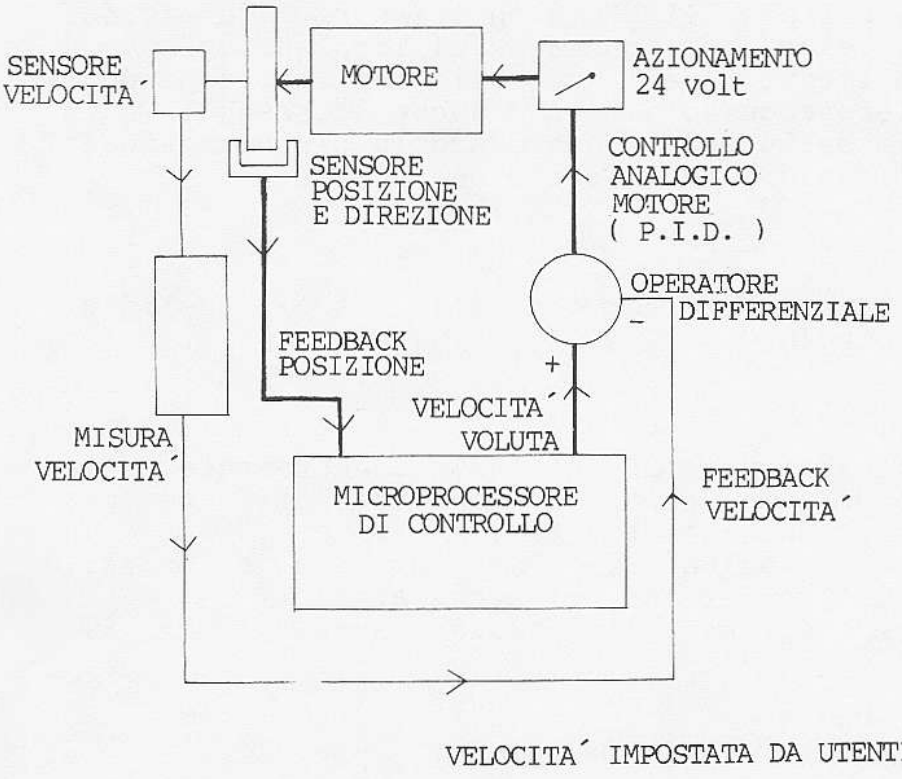


fig c.3.2

Ci sono due anelli (loop) di retroazione (feedback) che controllano contemporaneamente il movimento di ogni motore: uno in velocita´ e l´altro in posizione.

Il loop di controllo in velocita´ e´ realizzato analogicamente per assicurare la massima velocita´ di intervento.

Il loop di controllo in posizione e´ realizzato digitalmente attraverso un microprocessore, che ne garantisce la flessibilita´.
Lo schema che vedete (fig c.3.2) e´ uno schema di principio e non corrisponde alla complessa realta´ circuitale del controllo.

L´utente imposta una velocita´ (se vuole anche una variazione di velocita´ nel tempo, cioe´ un´accelerazione) comunicandola al microprocessore nella BASE.
Il microprocessore trasforma la velocita´ impostata in un segnale elettrico che tramite un operatore di controllo/pilotaggio motore porta il motore verso la velocita´ desiderata.
Un sistema di rilevamento tachimetrico misura la velocita´ effettivamente raggiunta dal motore .
L´operatore di controllo/pilotaggio confronta la misura di velocita´ effettuata con quella desiderata e apporta le opportune correzioni di pilotaggio al motore per ottenere la velocita´ voluta.

Contemporaneamete il microprocessore, con un sistema di sensori di posizione, misura lo spostamento dell´albero motore: ogni giro dell´albero motore corrisponde a 7 unita´ di conteggio.
Questa unita´ di conteggio e´ il minimo spostamento che la BASE e´ in grado di apprezzare: la chiamiamo "passo".
Nel paragrafo C.4 riporteremo questo "passo" sull´albero ad un "passo" di spostamento della ruota.

Il microprocessore confronta periodicamete (normalmente ogni 41,65 millisecondi, cioe´ 24 volte in un secondo) il conto dei passi effettivamete svolti con il numero dei passi previsti fino a quel momento dai parametri di moto che l´utente gli ha fornito.

Se ci sono ritardi o anticipi il microprocessore interviene correggendo il valore della velocita´ impostata sul loop analogico.
Se nella fase finale del movimento i passi contati non corrispondono con quelli aspettati, il microprocessore puo´ anche correggere il tempo di movimento impostato dal´utente, accorciandolo o prolungandolo entro certi limiti, pur di raggiungere la posizione voluta.

Chi volesse saperne di piu´ sui sistemi di controllo e sugli anelli di regolazione, consulti il capitolo G.

Questo sistema di controllo assicura un notevole adattamento alle condizioni operative dei motori, ma non puo´ operare efficacemente sotto una velocita´ minima di controllabilita´.
Se si scende sotto questa velocita´ il moto avviene a scatti, incidendo negativamente sulle prestazioni del sistema.
Tale velocita´ e´ di 1,36 cm/sec, per la marcia bassa, e di 5,5 cm/sec, per la marcia alta.

Riassumendo il range di velocita´ controllabili sulla BASE e´:

- da 1,36 a 6,6 cm/sec (marcia bassa)

- da 5,5 a 26,5 cm/sec (marcia alta)

C.4: Controlliamo i motori della BASE di MODULUS

A questo punto abbiamo in mano tutti gli ingredienti per controllare il movimento della BASE MODULUS: si tratta solo di "miscelarli" nel modo giusto per realizzare la ricetta.

1. DATI ESSENZIALI

Cosa vuole la BASE per essere comandata nel movimento?
Vuole sapere la "legge del moto monodimensionale", o legge oraria, da assegnare ad ogni motore ed il suo tempo di validita´.

La legge del moto e´ un´equazione che descrive come varia la posizione dell´oggetto mobile, o meglio di un suo punto di riferimento, nel tempo.
Usualmente la posizione varia nel piano o nello spazio, cioe´ in spazi matematici a 2 o 3 dimensioni.

Per noi e´ ancora piu´ semplice: MODULUS vuole sapere solo una legge del moto monodimensionale: che cioe´ descrive come varia nel tempo lo spazio percorso lungo la traiettoria dal punto di partenza.

Per esempio, se la ruota sinistra della BASE viaggia a velocita´ costante, allora possiamo descriverne il movimento lungo questi percorsi (vedi fig c.4.1) con la stessa legge oraria per la ruota sinistra, perche´ hanno tutti la stessa lunghezza.

Non e´ cosi´ per la ruota destra che segue in ognuno dei 3 esempi 3 leggi diverse.

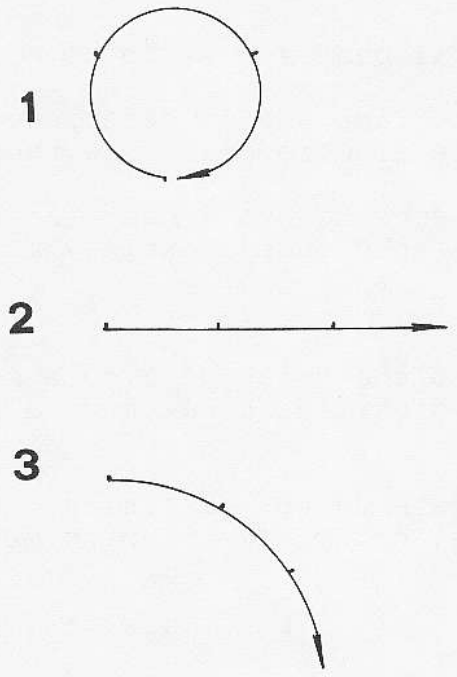


fig c.4.1

Se la velocita´ costante della ruota sinistra e´ di 5 cm/sec, la legge oraria, che da lo SPAZIO percorso a partire dall´inizio lungo la triettoria, e´:

SPAZIO (funzione del tempo) = 5 cm/sec * TEMPO

A seconda del valore di TEMPO che metto nell´equazione cambia il valore della variabile SPAZIO.

Questo descrive matematicamente il fatto che se osservo la BASE in moto a tempi diversi la trovo in posizioni diverse nello spazio di movimento.

Si dice cosi´ che SPAZIO e´ una funzione della varabile TEMPO.
In termini piu´ simbolici sarebbe:

$$s(t) = v * t$$

che esprime la legge del moto uniforme (= a velocita´ costante).

Piu´ in generale la nostra legge oraria e´ una funzione generica della variabile tempo:

$$s(t) = F(t)$$

Ma torniamo al caso specifico di MODULUS.

Per inciso notiamo che la direzione del moto noi la diamo descrivendo separatamente il moto delle due ruote destra e sinistra.

La BASE non accetta una legge oraria espressa da una funzione generica F(t), che sarebbe quasi inutile, ma solo questo tipo di legge:

$$s(t) = (A * t\hat{}2)/16 + (V * t)$$

Vi ricordiamo che usiamo "^" come simbolo di elevazione a potenza: per esempio t^2 significa t elevato alla seconda potenza, cioe´ t al quadrato.

Se vi ricordate un po´ di fisica questa legge e´ una variante di quello che si chiama "moto uniformemente accelerato", la cui equazione piu´ generale e´:

$$s(t) = (A * t\hat{}2)/2 + (V * t) + P$$

dove A e´ l´accelerazione
V e´ la velocita´ iniziale (al tempo t=0)
P e´ la posizione iniziale (al tempo t=0)

Se P = 0 allora si intende che si parte dalla posizione iniziale 0.
Possiamo sempre metterci d´accordo perche´ sia cosi´: basta chiamare 0 la posizione iniziale, qualunque essa sia.
In questo caso la legge del moto diventa:

$$s(t) = (A * t^2)/2 + (V * t)$$

NOTA
Se anche A = 0 abbiamo s(t) = V * t cioe´ moto uniforme a velocita´ costante V.
Se invece V = 0 abbiamo s(t) = (A * t^2)/2 cioe´ moto uniformemete accelerato (cioe´ il corpo accelera costantemete) partendo da velocita´ iniziale nulla.
Se A = V = 0 abbiamo s(t) = 0 : siamo fermi, velocita´ e accelerazione nulla.

L´equazione che usa MODULUS e´ proprio questa salvo dividere il termine di accelerazione per 8.

Quello che noi passiamo a MODULUS per descrivere il moto di ogni ruota sono i coefficienti A8 e V della

$$s(t) = (A8 * t^2)/16 + V * t$$

cioe´ l´accelerazione A moltiplicata per 8 (A8) e la velocita´ V

NOTA

L´accelerazione A, che si puo´ raggiungere nei casi concreti, e´ un numero piccolo e spesso dotato di virgola.

I parametri di moto di MODULUS sono numeri interi.
Se moltiplichiamo l´accelerazione per 8 diventa un numero piu´ grande e si puo´ tranquillamente sopprimere la virgola senza perdere troppo in precisione.

MODULUS dividera´ il valore di A8 che noi gli passiamo per 8, riottenendo cosi´ l´accelerazione A voluta.

Inoltre dovremo dare il tempo TM (Tempo di Movimento) di validita´ di questi due parametri.

Per esempio, se la ruota deve andare a V = 5 cm/sec per 32 secondi allora:

```
A8 = 0
 V = 5
TM = 32
```

Cio´ significa che la ruota seguira´ la legge oraria

$$s(t) = 5 * t$$

con t che va da 0 a 32, cioe´ 0 <= t <= TM

D´ORA IN POI CHIAMEREMO A8 E V PARAMETRI DEL POLINOMIO CHE ESPRIME LA LEGGE ORARIA PER MODULUS.
PER ESTENSIONE USEREMO IL NOME PARAMETRI/POLINOMIO PER TM E TUTTI GLI ALTRI DATI COMPLEMENTARI CHE ACCOMPAGNANO A8 E V, COME VEDREMO FRA POCO.

Fermiamoci un attimo e meditiamo sulle unita´ di misura di MODULUS.

2. UNITA´ DI MISURA

MODULUS non ragiona in secondi, centimetri o metri: usa delle unita´ di misura sue che occorre conoscere.
Questo non crea problemi per la conversione: bastano due conti che un programmino potra´ fare tranquillamente per voi.
Vediamo come.

- Il tempo si misura in UTA (Unita´ di Tempo di Aggiornamento), dove 1 UTA = 41,65 millisecondi, cioe´ 0,04165 secondi.

NOTA

Per i curiosi, 1 UTA e´ l´intervallo di tempo che intercorre tra un controllo di posizione ed il successivo.
In pratica il microprocessore, che gestisce il controllo di posizione del motore, va a confrontare la posizione reale con quella teorica desunta dalla legge oraria 24 volte al secondo, se 1 UTA = 41,65 millisecondi.

Vedremo nel paragrafo D.5 che e´ possibile cambiare il valore di 1 UTA.
Ma farlo con cognizione di causa e´ estremamente difficile.
SIRIUS non garantisce il buon funzionamento di MODULUS con valori diversi da quello standard.

TM puo´ variare da 0 a 255 UTA, cioe´ da 0 a 10,6 secondi.
Se volete che gli stessi parametri di movimento valgano per piu´ di 10,6 secondi basta ridarli a MODULUS nel successivo insieme di parametri.

- Lo spazio si misura IN UEP (Unita´ di Encoder Posizionale), dove 1 UEP = 0,014 cm (cambio su marcia bassa) o 1 UEP = 0,056 cm (cambio su marcia alta).

NOTA
Per i curiosi, 1 UEP e´ la distanza coperta da una ruota perche´ il suo sensore di posizione (montato sull´albero motore) conti un passo.
Questa unita´ corrisponde al minimo movimento distinguibile dal microprocessore che controlla i motori della BASE.
Il valore di 1 UEP e´ determinato da parametri meccanici (demoltipliche, diametro ruote, tipo di sensori) ed e´ quindi fisso.

I valori di 1 UEP che vi abbiamo dato sono quelli "ideali".

Il valore "reale" puo´ variare leggermente a causa delle tolleranze di produzione e dall´usura delle parti meccaniche.
Descriveremo nel paragrafo C.5.3 un possibile procedimento di calibrazione si queste costanti di conversione.

Per le normali applicazioni possiamo considerare:

1 UEP = 0,056 cm (marcia bassa)

1 UEP = 0,014 cm (marcia alta)

A questo punto basta eseguire le trasformazioni:

TEMPO (in UTA) = 24 * TEMPO (in secondi)

marcia bassa: - - -

SPAZIO (in UEP) = 70,42 * SPAZIO (in cm)

VELOCITA´ (in UEP/UTA) = 2,93 * VELOCITA´ (in cm/sec)

ACCELERAZIONE (in UES/UTA^2) = 0,12 * ACCELERAZIONE (in cm/sec^2)

marcia alta: - - -

SPAZIO (in UEP) = 17,6 * SPAZIO (in cm)

VELOCITA´ (in UEP/UTA) = 0,73 * VELOCITA´ (in cm/sec)

ACCELERAZIONE (in UES/UTA^2) = 0,03 * ACCELERAZIONE (in cm/sec^2)

ATTENZIONE: MODULUS vuole tutti i parametri V, A8, TM (e anche tutti gli altri che vedremo) come NUMERI INTERI.
I risultati delle trasformazioni, che sono numeri dotati di virgola, vanno arrotondati al piu´ vicino intero.
Un metodo consiste nel sommare 0,5 al numero e poi considerare solo la parte intera.
Per esempio: 6,6 diventa 7; 88,3 diventa 88; 53,5 diventa 54; 53,56 diventa 54;...

V ed A8 oltre ad essere interi sono dotati di segno (motore avanti +, motore indietro -) e devono essere compresi fra -127 e +127.
TM invece accetta valori interi compresi fra 0 e 255.

ATTENZIONE: In pratica V e A8 hanno un range (arco di variabilita´) molto piu´ limitato.
Cio e´ dovuto da una parte a limiti meccanici (MODULUS non e´ un´auto da corsa) e dall´altra a limiti elettronici di controllo motori.
I valori di V ed A8 entro cui la BASE risponde correttamente sono da -28 a -4 e da +4 a +28 (notate che sono esclusi i valori -3,-2,-1,0,+1,+2,+3).

3. DATI COMPLEMENTARI

Abbiamo diviso questi dati/parametri da quelli essenziali perche´ questi non descrivono il moto di per se, ma forniscono dei parametri necessari allo specifico meccanismo di controllo microelettronico del moto della BASE.

Vediamo da dove sorge la necessita´ di questi parametri.

1. Supponiamo di voler fare eseguire al robot questa traiettoria.

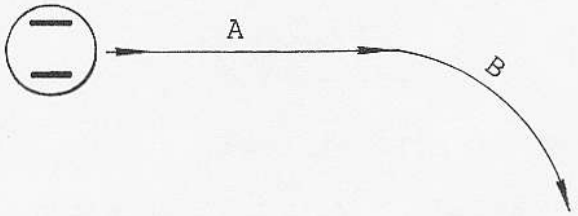


fig c.4.2

Serviranno due insiemi di parametri: uno per il percorso A e l´altro per il percorso B.
MODULUS offre due modi di realizzare il percorso A + B: separati o raccordati.
Se i percorsi sono separati, allora la BASE , arrivata in fondo al percorso A si ferma e poi riparte per il percorso B.
Viceversa, nel modo raccordato, i 2 insiemi di parametri di moto vengono eseguiti senza passare attraverso le fermate.
Dovremo quindi specificare se un gruppo di parametri e´, o meno, raccordato con il successivo.

Vedremo fra poco che la fermata puo´ creare problemi di precisione. Per questo, se serve un movimento molto preciso, come per i disegni con la penna-plotter, e´ consigliabile usare insiemi di parametri raccordati fra loro e minimizzare le fermate.

2. Osserviamo che il fatto che i parametri fondamentali V,A e TM siano numeri interi puo´ creare dei problemi.

Come facciamo, per esempio, a far fare alla BASE un percorso di questo tipo, usando la marcia alta e una velocita´ di 20 UEP/UTA (27,4 cm/sec) ?

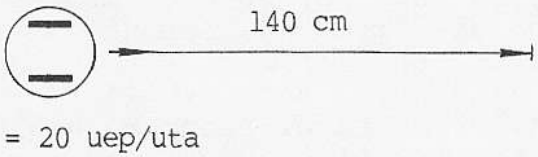


fig c.4.3

Il percorso e´ di 145 cm, ovvero 2552 UEP (marcia alta).
Il tempo TM necessario per il percorso e´:

TEMPO TM = SPAZIO/VELOCITA´ = 2552/20 = 127,6 UTA

Se scegliamo TM = 127 allora lo spazio che la BASE percorrera´ sara´

SPAZIO = VELOCITA´ * TEMPO = 20 * 127 = 2540 UEP

c´e´ una differenza di -12 UEP, cioe´ -0,68 cm, con la distanza desiderata.
D´altro canto, se scegliamo TM = 126 otteniamo una differenza di +8 UEP, cioe´ +0,45 cm, sulla distanza desiderata, e la situazione non migliora sostanzialmente.

Queste differenze possono sembrare piccole, ma se si effettuano successivi movimenti gli errori si sommano e possono diventare decisamente grandi.

Per rimediare a questo la BASE MODULUS offre due parametri.

1) la POSIZIONE FINALE, ovvero lo spostamento totale che desideriamo far compiere alla BASE con quell´insieme di parametri di moto.
Nell´esempio sopra PF = 1232 UEP, anzi PF = +1232 UEP (perche´ il movimento e´ in avanti).

2) il TEMPO FINALE, ovvero il tempo prima della fine del movimento in cui iniziare il controllo del raggiungimento della posizione finale.

In sostanza la BASE, TF UTA prima dello scadere del tempo TM, inizia a confrontare (24 volte al secondo) se si sta avvicinando alla posizione finale volUTA PF.
Quando la posizione finale e´ stata raggiunta la BASE disattiva i motori, anche se il tempo di movimento TM non e´ ancora finito.
Viceversa, se allo scadere del tempo TM la posizione finale non e´ stata raggiunta, la BASE prosegue il movimento oltre il tempo TM (per un massimo di circa 4 secondi) pur di raggiungere la posizione finale.

Questo meccanismo permette di recuperare gli errori di troncamento dovuti ai parametri interi, ma ovviamente serve anche a recuperare gli errori veri e propri nel movimento.
Vedremo poi che MODULUS vi fornira´ anche, se richiedete il messaggio di STATO BASE, l´errore di spostamento alla fine del movimento.

Ovviamente deve essere TF <= TM.
Di solito un valore di TF = 4 UTA va bene.
Se si pone TF = 0 allora il controllo sulla posizione finale inizia solo quando e´ scaduto il tempo di movimento TM.
Vale la pena di aumentare il valore di TF solo se si ha motivo per credere di aver viaggiato piu´ velocemente del previsto.

La specifica di TF ha senso solo se il movimento specificato dai parametri NON E´ RACCORDATO, e cioe´ se alla fine del movimento la BASE si deve fermare.

Vediamo ora il problema della fermata.
Questo non creera´ necessita´ di introdurre nuovi parametri, ma val la pena di discuterlo ora, perche´ e´ chiaramente legato ai problemi che abbiamo appena visto.

Il problema della fermata e´ prodotto dal fatto che fermarsi significa decelerare, e decelerare significa avere a che fare con l´inerzia del corpo in movimento.
Piu´ la decelerazione e´ brusca, cioe´ rapida, e tanto maggiore e´ la forza che tende a far proseguire avanti la BASE.

Questa forza provoca due fenomeni: da una parte lo slittamento delle ruote e dall´altra l´avanzamento, senza slittamento, della BASE anche a motore disattivato.

Il primo fenomeno e´ irrecuperabile: la BASE non e´ dotata di "occhi" e regola il suo moto sul conto dei passi (delle UEP) intesi come frazioni di giri della ruota.
Se la ruota slitta, e dunque gira ma non avanza (slitta) o avanza senza girare (scivola), la BASE sbaglia il conteggio dei "passi" e, se non ha altri mezzi per "fare il punto", perde letteralmente la "bussola".

Lo slittamento si puo´ evitare usando la BASE su superfici non troppo lisce (non su pavimenti molto scivolosi, ad esempio) e contemporaneamenteo salde (non sulla ghiaia o su fondi sabbiosi).

Il secondo fenomeno e´ invece recuperabile: la BASE avanzera´ inevitabilmente ancora un poco a motori disattivati, ma siccome le ruote girano l´avanzamento sara´ registrato e contato dal microprocessore di controllo.
Richiedendo successivamente lo stato della BASE (vedi paragrafo D.5), la BASE restituisce gli errori di posizione finale commessi.
Sapremo cosi´ di quanto (quanti UEP) si e´ andati avanti oltre il punto previsto e potremo regolarci di conseguenza.

In ogni caso la regola principe per non slittare o per fermarsi in poco spazio, come ben sanno i guidatori, e´ quella di decelerare con dolcezza e arrivare piano alla "frenata finale".

Vediamo qualche idea per fermarsi con precisione.

IDEA DI FERMATA 1

Un certo tempo prima della fermata, lo chiameremo TD, si imposta un insieme di parametri di moto che faranno decelerare gradualmente la BASE fino alla minima velocita´ operativa (+/- 4 UEP/UTA).
Si da poi un altro insieme di parametri che mantengano in ultimo la minima velocita´ (VU = +/- 4 UEP/UTA) per gli ultimi istanti (per un tempo TS) del tempo di movimento percorso, e poi si lascia fermare la BASE.
Imporremo che lo spazio percorso con la decelerazione e il tratto a velocita´ ci faccia arrivare sul punto finale voluto.

In pratica nel programmare normalmente il movimento si usano parametri/polinomi raccordati e si lascia per ultimo un solo insieme di parametri/polionomio non raccordato.
Per ottenere una fermata "dolce" e precisa e´ opportuno sostituire l´ultimo polinomio, quello non raccordato, con 3 polinomi/parametri, 2 dei quali raccordati e l´ultimo no.

Il primo dei tre polinomi realizza il vostro moto finale originariamente desiderato (= senza decelerazione) ma per un tempo inferiore.
Il secondo descrive la decelerazione dalla velocita´ V del originario ultimo polinomio a quella VU del tratto finale a velocita´ minima.
Il terzo, non raccordato, descrive il tratto finale a velocita´ minima VU per TU UTA prima della fermata.

Calcoleremo i parametri/polinomio dei tre movimenti in modo che lo spazio percorso con l´insieme dei tre movimenti sia uguale allo spazio che si sarebbe percorso con il vostro polinomio finale originario.

Descriviamo la procedura in simboli: supponiamo che il vostro ultimo tratto sia a velocita´ costante V (A8 = 0).

PARAMETRI FINALI ORIGINARI		NUOVI PARAMETRI :	1	2	3
A8:	0	A8:	0	-D	0
V:	V	V:	V	V	VU
TM:	TM	TM:	TN	TD	TU
PF:	PF	PF:	-	-	PU
raccordo:	no	raccordo:	si	si	no

V,TM e PF li avete stabiliti voi. VU,TU,D li stabiliamo di "ufficio": VU = +/- 4 , TU = 10, D = +/- 4 (marcia alta) o D = +/- 8 (marcia bassa).
Dunque PU = VU * TU = 40 UEP
Fatto questo i valori di TD e TN si determinano con le formule:

$$TD = (8 * (V - VU)) / D$$

$$TN = (PF + ((D/16) * TD^2) - (V * TD) - PU) / V$$

I valori scelti di TU e VU assicurano un tratto finale di 0,416 secondi a velocita´ minima (1,36 cm/sec con marcia bassa o 5,5 cm/sec con marcia alta).
Il valore di D scelto corrisponde a una decelerazione di 1 UEP/UTA^2 (8,19 cm/sec^2) con marcia bassa o di 0,5 UEP/UTA^2 (16,37 cm/sec^2) con marcia alta.
La decelerazione D e´ stata scelta di valore piu´ elevato con la marcia bassa, che da minori problemi di fermata.
Il segno +/- che abbiamo indicato davanti al valore di VU e D dipende dal senso di marcia: se il vostro ultimo tratto di moto e´ in avanti allora il segno e´ +, se e´ indietro allora e´ - .

Facciamo un esempio per chiarire l´uso del procedimento. Supponiamo che il nostro ultimo tratto di moto sia descritto dai seguenti parametri:

```
marcia alta  V:  25 ( 34 cm/sec )
            A8:  0
            TM:  48 ( 2 sec )
            PF:  1200 ( 68 cm )
```

calcoliamo:

```
TD = ( 8 * ( 25 - 4 )) / 4 = 42 UTA ( 1,75 sec )

TN = ( 1200 + ((4/16) * 42^2) - (25 * 42) - 40 ) / 25 =
   = 22,04 = 22 UTA ( 0,91 sec )
```

e quindi i 3 insiemi di parametri saranno:

```
                1   2   3

         A:     0  -4   0
         V:    25  25   4
        TM:    22  42  10
        PF:     -   -  40
```

Notate che il medesimo spazio viene compiuto in un tempo TM = TN + TD + TU = 74 UTA, ovviamente maggiore dei 48 originari.

Tre osservazioni ancora.

Prima: il medesimo discorso, ma rovesciato, si puo´ fare alla partenza, dove pero´ esiste solo il problema dell´eventuale slittamento e non quello della fermata oltre la posizione volUTA.
Anche se la velocita´ che abbiamo impostato alla partenza non viene raggiunta immediatamente la BASE e´ in grado di recuperare tramite il controllo di posizione finale descritto in questo stesso paragrafo, a patto di non avere slittamenti.

Seconda: vale la pena di complicarsi la vita con accelerazioni e decelerazioni solo quando si opera con la marcia alta, e quindi a velocita´ elevate, a meno che non si usi la BASE per disegnare: in tal caso il problema, data la precisione richiesta, sussiste tangibilmente anche per la marcia bassa.

In molte altre occasioni ci aiutera´ piu´ efficacemente, nel fare il punto, la conoscenza dell´ambiente combinata con l´azione di altri sensori: quelli di urto nella BASE o gli altri sensori applicabili a MODULUS dotato di spicchioteca.

Terza: usate questi consigli con buon senso.
Per esempio, non avete notato nulla di strano nell´esempio di "fermata" che abbiamo calcolato sopra?
Secondo voi i valori scelti e calcolati sono sensati?

Per esempio tenete conto che la velocita´ di scambio dati sul canale di comunicazione con la BASE e´ limitata (circa 30 caratteri al secondo piu´ i tempi di attesa introdotti dall´esecuzione del programma sul computer di controllo) e che quindi non ha senso passare alla BASE tanti polinomi raccordati con tempi di movimento piccoli: si rischia di non riuscire ad inviarli in tempo.

Facciamo due conti: a 300 baud 1 carattere viene trasmesso in circa 33 millisecondi; come vedremo nel capitolo D, un pacchetto contenente un polinomio e´ lungo 10 caratteri e poi abbiamo il pacchetto di risposta della BASE di 4 caratteri; in totale la comunicazione occupa 14 * 33 millisecondi = 0,46 sec = 11,2 UTA .
In realta´ la situazione e´ anche peggiore perche´ noi abbiamo supposto nulli i "ritardi di reazione" introdotti dal computer di controllo e dalla BASE, mentre in realta´ cio´ non e´ vero.

Ora, mandare dei polinomi/parametri con tempi di esecuzione di 10, ma anche 22, UTA non ha senso pratico rispetto ai tempi di comunicazione dei parametri stessi.

Morale: se usate movimenti raccordati, usate tempi di movimento sufficientemente lunghi (TM > 24 UTA).

IDEA DI FERMATA 2

Altra tecnica, piu´ intelligente ed efficiente ma piu´ laboriosa, e´ quella di, una volta raggiunto il punto voluto, eseguire una vera e propria frenata comandando per un brevissimo tempo il motore ad andare nel senso opposto.
Questo generera´ una forza che si sommera´ agli attriti nel fermare la BASE.
Si intuisce che usando questo metodo si rischia facilmente di far slittare la ruota e quindi di perdere l´affidabilita´ sul conteggio dello spazio percorso.

Inoltre il calcolo della velocita´ e/o accelerazione negative da applicare non si puo´ fare sulla carta, conoscendo solo i nostri parametri di moto.
Questa tecnica richiede degli adattamenti sperimentali alle condizioni operative come ad esempio tolleranza dei parametri elettrici nei motori e nell´elettronica di controllo, aderenza delle ruote al suolo, ...
Per poterla usare occorre dunque andare per tentativi.

Potete comandare la BASE, con la marcia veloce e a velocita´ V, per farle percorrere uno spazio S.
Alla fine del movimento richiedete alla BASE il messaggio di stato BASE che, come vedremo nel paragrafo D.5, contiene anche l´errore di posizione finale commesso; provate ora a raccordare al vostro ultimo polinomio un polinomio che imposti una velocita´ VO in senso opposto per un tempo TO e ripetete la prova variando VO e TO fino a annullare o minimizzare l´errore di posizione.
Ricordate di non usare VO troppo elevate e di controllare che non ci siano slittamenti visibili, altrimenti l´errore di posizione rilevato dalla BASE non e´ piu´ affidabile.
Se avete pazienza potete, invece di richiedere alla BASE l´errore di posizione, misurarlo voi valUTAndo la distanza coperta effettivamente dalla BASE.
La cosa va ripetUTA per varie velocita´ (es. 5 ,10, 15, ...) e per entrambe le marce, in quanto i valori di VO e TO necessari per frenare saranno diversi.

IDEA DI FERMATA 3

Un metodo, piu´ pragmatico, ma simile al precedente: potete comandare la BASE, con la marcia veloce, in modo da farle percorrere uno spazio S a velocita´ V.

Alla fine del movimento richiedete alla BASE il messaggio di stato BASE che, come vedremo nel paragrafo D.5, contiene anche l´errore di posizione finale commesso; annotate l´errore e ripetete per varie velocita´ (es. 5, 10, 15, 20, ...).
Ripetere lo stesso per la marcia bassa.
A questo punto calcolate tutti gli errori in percentuale e fatevi una tabella delle percentuali di errore relative ad ogni velocita´ di ogni marcia
Quando dovrete costruire il vostro polinomio che descrive l´ultimo tratto toglierete alla distanza da percorrere una percentuale pari a quella annotata nella tabella degli errori in "eccesso" in corrispondenza della velocita´ che intendete tenere.

Riassumiamo ora tutti i parametri che dobbiamo stabilire per il movimento della BASE MODULUS.

V: velocita´ in UEP/UTA (da +/- 4 a +/- 28)

A8: accelerazione in UEP/UTA^2 (da +/- 4 a +/- 28)

TM: tempo di movimento in UTA (da 0 a 255)

TF: tempo di controllo posizione finale in UTA (da 0 a 127)
(se il movimento non termina, ma prosegue con un altro movimento descritto da un polinomio raccordato con questo, allora TF non conta e di solito si pone uguale a 0)

PF: punto finale da raggiungere col movimento, in UEP (da 0 a +/- 32767)

RACCORDO: se i parametri sono raccordati allora la BASE non si ferma e prosegue il movimento eseguendo il polinomio seguente; se non sono raccordati, alla fine del movimento la BASE si ferma.
Il raccordo si indica ponendo un segno - al valore di TF. Viceversa, se non c´e´ raccordo, TF ha segno +.

In pratica la cosa e´ molto piu´ semplice di quel che sembra.
Se dobbiamo, per esempio, descrivere un movimento in avanti di 1 metro alla velocita´ di 15 cm/sec (con la marcia alta), facciamo i seguenti conti:

V = +15 cm/sec = +10,95 = +11 UEP/UTA

A8 = = 0

TF = 4 UTA

PF = +100 cm = +1760 UEP

TM = 1760/11 = 160 UTA

no raccordo = TF positivo

La marcia e´ avanti: dunque i due motori andranno pilotati con gli stessi parametri.

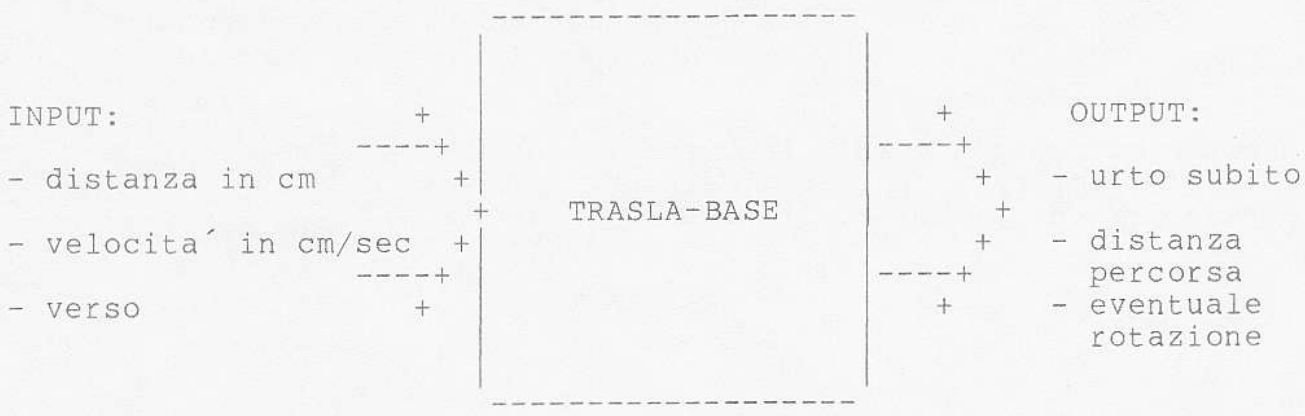
PARAMETRI MOTORE DESTRO: +5,0,160,+4,+1760
PARAMETRI MOTORE SINISTRO: +5,0,160,+4,+1760

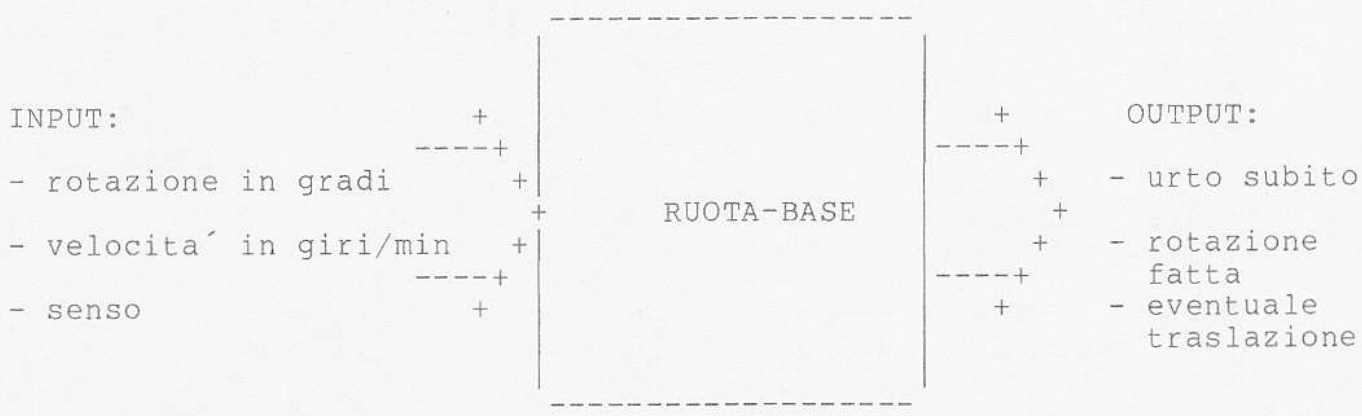
ESERCIZIO: traducete i movimenti degli esempi del capitolo C.1 (fig c.1.16 e c.1.17)in parametri/polinomi per i motori della BASE.

C.5.1: Rotazioni e traslazioni

Abbiamo visto nel paragrafo C.1 come realizzare una qualsiasi traiettoria approssimandola con delle spezzate.
Abbiamo anche visto che le spezzate si realizzano utilizzando unicamente traslazioni (moto rettilineo) e rotazioni (moto di rotazione attorno al centro della BASE).
Vogliamo ora progettare delle procedure che noi richiameremo all'interno dei nostri programmi per comandare il movimento della BASE.

Avremo bisogno di due procedure: una TRASLA e una RUOTA.
Ipotizziamo di usare velocita' costante; ci penseranno le procedure stesse ad introdurre a fine traslazione o rotazione una opportuna decelerazione, secondo il primo metodo descritto nel paragrafo C.4 .
Questo significa che avremo una fermata alla fine di ogni rotazione o traslazione.





Siccome la realizzazione non e' semplice, scomponiamo il problema in sottoproblemi.

Useremo d´ora in poi una notazione meno grafica ma piu´ precisa dal punto di vista algoritmico: useremo delle strutture di controllo di programma pseudo-PASCAL che hanno il pregio di essere chiare.
Nel paragrafo E.1 troverete il metodo per tradurle in strutture di controllo BASIC o di altri linguaggi.

Detto questo esaminiamo un po´ piu´ in dettaglio le due procedure:
NB: dx = destra e sx = sinistra

```
trasla_BASE (IN:dist.,vel.,verso;OUT:urto,dist._coperta,rotaz.);

BEGIN

calcola_parametri_traslaz.(IN:dist.,vel.,verso;OUT:polinomi_dx&sx);

    muovi_BASE(IN:polinomi_dx&sx;OUT:urto,errori_dx&sx);

    calcola_errori(IN:errori_dx&sx;OUT:err_distanza,err_angolo);

    dist._coperta := dist. -  err_distanza;

    rotaz. := err_angolo;

    OUTPUT urto,dist._coperta,rotaz.;
END;
```

```
ruota_BASE (IN:gradi,vel.,senso;OUT:urto,rotaz._fatta,traslaz.);

BEGIN
    calcola_parametri_rotaz.(IN:gradi,vel.,senso;OUT:polinomi_dx&sx);

    muovi_BASE(IN:polinomi_dx&sx;OUT:urto,errori_dx&sx);

    calcola_errori(IN:errori_dx&sx;OUT:err_distanza,err_angolo);

    rotaz._fatta := gradi -  err_angolo;

    traslaz. := err_distanza;

    OUTPUT urto,rotaz._fatta,traslaz.;
END;
```

Le due procedure sono progettate in modo da utilizzare il piu´ possibile delle sottoprocedure (sottoprogrammi) comuni.
Infatti notate che la transla_BASE e la ruota_BASE sono uguali a meno della prima sottoprocedura di calcolo parametri.

Le procedure di calcolo dei parametri accetteranno in ingresso i dati di rotazione o traslazione espressi usando come unita´ centimetri, gradi, secondi e produrranno in uscita una serie di polinomi raccordati (tranne gli ultimi) per il motore dx e di sx.
I parametri in ingresso non devono essere interi e non ci sono limitazioni di distanza o tempo: ci pensera´ la procedura di calcola parametri a gestire le approssimazioni dovute al fatto che i polinomi sono fatti con parametri interi e a produrre piu´ polinomi uguali se il tempo di movimento e´ piu´ lungo di quello permesso da un unico polinomio.

La procedura di muovi_BASE provvede ad inviare alla BASE i parametri/polinomio e a seguire il corso del movimento.
Alla fine del movimento richiedera´ alla BASE il messaggio di stato da cui ricavare le indicazioni di urto e gli errori di posizione residui dei due motori.

Se intendete comandare il movimento della BASE con traiettorie qualsiasi, cioe´ non utilizzando solo rotazioni e traslazioni, o se non volete le fermate introdotte dalle procedure di ruota_BASE e trasla_BASE potete allora usare direttamente questa procedura, a patto di darle in ingresso una lista di parametri/polinomi per i motori dx e sx.

La procedura di calcola_errori non fa altro che interpretare gli errori di posizione finale restituiti dalla muovi_BASE e tradurli in termini di centimetri di spostamento e gradi di rotazione non voluti, ma effettuati nel movimento.

Potrete cosi´ tenere il punto della navigazione del robot ed eventualmente correggere la "rotta" con traslazioni e/o rotazioni successive.

Se vi interessa il dettaglio delle procedure proseguite leggendo il paragrafo C.5.2, se invece pensate a qualcosa di meno laborioso e gia´ pronto, passate a leggere il paragrafo E.2 dove sono descritte le istruzioni di BASIC introdotte da SIRIUS per il controllo del robot.

Le procedure che vi descriveremo nel paragrafo C.5.2., C.5.3, D.7 ed E.1 sono invece dirette a chi intende costruirsi i programmi di controllo per MODULUS.
Ovviamente il problema non ha un unica soluzione e quella che vi indichiamo e´ solo una delle tante possibili.

AVVERTIAMO CHE QUESTE PROCEDURE SONO CORRETTE, MA SIRIUS GARANTISCE ED ASSISTE SOLO IL SOFTWARE ACQUISTATO SU SUPPORTI MAGNETICI O ELETTRONICI CON MARCHIO SIRIUS.

C.5.2: Come scriverle in un programma

Riprendiamo i flussi delle trasla_BASE e ruota_BASE e introduciamo accanto alle strutture di controllo pseudo-PASCAL delle notazioni di funzioni e di variabili pseudo-BASIC (ad esempio una variabile intera termina con "%", una stringa con "$" e le altre sono reali, ecc).

```
trasla_BASE (IN:cm,vel,verso%;OUT:urto%,traslaz,rotaz);

BEGIN
    calcola_parametri_traslaz(IN:cm,vel,verso;OUT:poli_dx$,poli_sx$);

    muovi_BASE(IN:poli_dx$,poli_sx$;OUT:urto%,errore_dx%,errore_sx%);

    calcola_errori(IN:errore_dx%,errore_sx%;OUT:err_distanza,
                   err_angolo;

    traslaz := cm -  err_distanza;

    rotaz := err_angolo;

    OUTPUT urto%,traslaz,rotaz;
END;



ruota_BASE (IN:gradi,vel_rot,senso%;OUT:urto%,rotaz,traslaz);

BEGIN
    calcola_parametri_rotaz.(IN:gradi,vel_rot,senso%;OUT:poli_dx$,
                             poli_sx$);

    muovi_BASE(IN:poli_dx$,poli_sx$;OUT:urto%,errore_dx%,errore_sx%);

    calcola_errori(IN:errore_dx%,errore_sx%;OUT:err_distanza,
                   err_angolo);

    rotaz := angolo -  err_angolo;

    traslaz := err_distanza;

    OUTPUT urto%,rotaz,traslaz;
END;
```

Descriviamo i parametri di ingresso a queste due procedure:

- cm : e´ la distanza in centimetri da percorrere in linea retta
- vel : e´ la velocita´, in cm/sec, da tenere nel percorso
- verso%: 0 = avanti, 1 = indietro

- gradi : e´ la rotazione da effetture espressa in gradi: 1 giro = 360 gradi
- vel_rot : e´ la velocita´ di rotazione espressa in giri/min, per praticita´
- senso% : 0 = rotazione oraria, 1 = rotazione antioraria

- urto% : 0 = nessun urto subito e movimento terminato senza anomalie, da 1 a 254 = movimento interrotto per urto, 255 = movimento interrotto per altre anomalie.
  Nel caso di urto il valore di urto% corrisponde a quello del byte bumpers nel messaggio di stato BASE (vedi paragrafo C.2).
- rotaz : rotazione in gradi effettivamentre subita dalla base a fine movimento; se il numero e´ positivo la rotazione e´ oraria
- traslaz : distanza in centimetri effettivamente coperta dalla BASE a fine movimento; se il numero e´ positivo lo spostamento e´ in avanti

Inoltre si suppone che voi abbiate definito una variabile globale (vedi paragrafo E.1) di nome marcia% che contiene 0 se la marcia selezionata sul cambio e´ bassa o 1 se la marcia selezionata e´ quella alta.
Altro parametro/scelta fisso da definire e´ decelerazione% : se volete la decelerazione graduale a fine movimento allora assegnate alla varabile globale decelerazione% il valore 1.

Cominciamo con la calcola_parametri_traslaz.
Introdurremo due sottoprocedure per il calcolo della decelerazione e per la costruzione di un "comando di invia polinomio" secondo il formato previsto dal livello applicativo del protocollo di comunicazione con la BASE (vedi paragrafo D.5).
Queste sottoprocedure verranno descritte immediatamente dopo.

Naturalmente si possono introdurre molte altre sottoprocedure, in quanto la calcola_parametri_traslaz e la calcola_parametri_rotaz sono molto simili.
In ogni caso questa che vi diamo e´ solo una traccia per realizzare i programmi di controllo della BASE: sarete voi che deciderete il modo migliore per organizzare e scrivere i vostri programmi.

```
calcola_parametri_traslaz(IN:cm,vel,verso%;OUT:poli_dx$,poli_sx$)

BEGIN
    alta% := 1;
    bassa% := 0;
    raccordo% := 240;
    stop := 4;
    polinomi$ := stringa_vuota$;
    vera := 1;

    IF ( marcia% = alta% )
    THEN BEGIN
            conv_spazio := 17,6;
            conv_velocita´ := 0,73;
            conv_accelerazione := 0,03;
    END;

    ELSE BEGIN
            conv_spazio := 70,42;
            conv_velocita´ := 2,93;
            conv_accelerazione := 0,12;
    END;

    conv_tempo := 24;

    spazio% := conv_spazio * cm;
    velocita´% := conv_velocita´ * vel;
    IF ( velocita% > 28 )
    THEN BEGIN
            velocita% := 28;
            vel := velocita% / conv_velocita´;
    END;
    tempo% := conv_tempo * ( cm / vel );

    WHILE ( tempo% > 255 )
    DO BEGIN
          spazio255% := velocita´% * 255;
          produci_poli(IN:verso%,0,velocita´%,raccordo%,255,0
                       ;OUT:poli$);
          polinomi$ := polinomi$ + poli$;
          tempo% := tempo% - 255;
          spazio% := spazio% - spazio255%;
    END;
```

```
    A% := 0;
    V% := velocita%;
    TF% := stop%;
    TM% := tempo%;
    PF% := spazio%;

    IF ( decelerazione% = vera% )
    THEN BEGIN
            calcola_decelerazione(IN:A%,V%,TM%,PF%;
                                  ;OUT:D%,VU%,TN%,TD%,TU%,PU%)
            produci_poli(IN:verso%,0,V%,raccordo%,TN%,0;OUT:poli$);
            polinomi$ := polinomi$ + poli$;
            produci_poli(IN:verso%,-D%,V%,raccordo%,TD%,0
                         ;OUT:poli$);
            polinomi$ := polinomi$ + poli$;
            produci_poli(IN:verso%,0,VU%,raccordo%,TU%,PU%
                         ;OUT:poli$);
            polinomi$ := polinomi$ + poli$;
    END;

    ELSE BEGIN
            produci_poli(IN:verso%,A%,V%,TF%,TM%,PF%;OUT:poli$);
            polinomi$ := polinomi$ + poli$;
    END;

    poli_dx$ := polinomi$;
    poli_sx$ := polinomi$;

    OUTPUT: poli_dx$,poli_sx$;
END;
```

Ora vediamo la calcola_parametri_rotaz.

```
calcola_parametri_rotaz(IN:gradi,vel_rot,senso%;OUT:poli_dx$,
                        poli_sx$)

BEGIN
    alta% := 1;
    bassa% := 0;
    raccordo% := 240;
    stop := 4;
    poli_dx$ := stringa_vuota$;
    poli_sx$ := stringa_vuota$;
    vera := 1;
    avanti% :=0;
    indietro :=1;
    orario% := 0;

    IF ( marcia% = alta% )
    THEN BEGIN
            conv_spazio := 17,6;
            conv_velocita´ := 0,73;
            conv_accelerazione := 0,03;
    END;

    ELSE BEGIN
            conv_spazio := 70,42;
            conv_velocita´ := 2,93;
            conv_accelerazione := 0,12;
    END;

    conv_tempo := 24;

    spazio_cm := ( gradi * 27 * 3,14 ) / 360;
    spazio% := conv_spazio * spazio_cm;
    vel_ruote := ( 6,28 * 27 * vel_rot ) / 60;
    velocita´% := conv_velocita´ * vel_ruote;
    IF ( velocita% > 28 )
    THEN BEGIN
            velocita% := 28;
            vel_ruote := velocita% / conv_velocita´;
    END;
    tempo := ( gradi * 6,28 * 27 ) / ( vel_ruote * 360 );
    tempo% := conv_tempo * tempo;

    IF ( senso% := orario% )
    THEN BEGIN
            verso_sx% := avanti%;
            verso_dx% := indietro%;
    END;

    ELSE BEGIN
            verso_sx% := indietro%;
            verso_dx% := avanti%;
    END;
```

```
    WHILE ( tempo% > 255 )
    DO BEGIN
          spazio255% := velocita´% * 255;
          produci_poli(IN:verso_sx%,0,velocita´%,raccordo%,255,0
                       ;OUT:poli$);
          poli_sx$ := poli_sx$ + poli$;
          produci_poli(IN:verso_dx%,0,velocita´%,raccordo%,255,0
                       ;OUT:poli$);
          poli_dx$ := poli_dx$ + poli$;
          tempo% := tempo% - 255;
          spazio% := spazio% - spazio255%;
    END;

    A% := 0;
    V% := velocita%;
    TF% := stop%;
    TM% := tempo%;
    PF% := spazio%;

    IF ( decelerazione% = vera% )
    THEN BEGIN
            calcola_decelerazione(IN:A%,V%,TM%,PF%,
                                  ;OUT:D%,VU%,TN%,TD%,TU%,PU%)
            produci_poli(IN:verso_sx%,0,V%,raccordo%,TN%,0
                         ;OUT:poli$);
            poli_sx$ := poli_sx$ + poli$;
            produci_poli(IN:verso_dx%,0,V%,raccordo%,TN%,0
                         ;OUT:poli$);
            poli_dx$ := poli_dx$ + poli$;
            produci_poli(IN:verso_sx%,-D%,V%,raccordo%,TD%,0
                         ;OUT:poli$);
            poli_sx$ := poli_sx$ + poli$;
            produci_poli(IN:verso_dx%,-D%,V%,raccordo%,TD%,0
                         ;OUT:poli$);
            poli_dx$ := poli_dx$ + poli$;
            produci_poli(IN:verso_sx%,0,VU%,raccordo%,TU%,PU%
                         ;OUT:poli$);
            poli_sx$ := poli_sx$ + poli$;
            produci_poli(IN:verso_dx%,0,VU%,raccordo%,TU%,PU%
                         ;OUT:poli$);
            poli_dx$ := poli_dx$ + poli$;
    END;

    ELSE BEGIN
            produci_poli(IN:verso_sx%,A%,V%,TF%,TM%,PF%;OUT:poli$);
            poli_sx$ := poli_sx$ + poli$;
            produci_poli(IN:verso_dx%,A%,V%,TF%,TM%,PF%;OUT:poli$);
            poli_dx$ := poli_dx$ + poli$;
    END;

    OUTPUT: poli_dx$,poli_sx$;
END;
```

Ora vediamo le sottoprocedure calcola_decelerazione e produci_poli.

```
calcola_decelerazione(IN:A%,V%,TM%,PF%,;OUT:D%,VU%,TN%,TD%,TU%,PU%)

BEGIN
    VU% :=4;
    TU% :=10;
    alta% := 1;

    IF ( marcia% = alta% )
    THEN BEGIN
            D% := 4;
    END;

    ELSE BEGIN
            D% := 8;
    END;

    PU% := VU% * TU%;
    TD% := ( 8 * ( V% - VU% ) / D% );
    TN% := (PF% + ((D%/16) * TD%^2 ) - (V% * TD%) - PU%) / V%

    IF ( TN% < 0 )
    THEN BEGIN
            TD% := TM% - TU%;
            VU% := V% - ( D% * TD% / 8 )
            TN% := 0;
    END;

    OUTPUT: D%,VU%,TN%,TD%,TU%,PU%;
END;
```

Notate che per TU% abbiamo usato un valore di soli 10 UTA, malgrado le raccomandazioni del paragrafo C.4 di non usare tempi di movimento troppo piccoli.
Qui pero´ il problema non sussiste, in quanto TU e´ il tempo dell´ultimo movimento prima della fermata, e quindi non ci sara´ il problema di comunicare i successivi parametri in tempo.

La produci_poli, che vediamo adesso, non fa altro che preparare i parametri nel formato voluto dal protocollo di comunicazione con la BASE (vedi paragrafo D.5).
Anticipiamo solo che i numeri negativi si codificano mettendo a 1 il bit piu´ significativo.
Nel caso di un numero che sta in un byte, ad esempio, cio´ significa fare un OR con 240 dec (80 hex, 10000000 bin).
Cio´ implica che in un byte potremo codificare numeri da -127 a +127.

```
produci_poli(IN:verso%,A8%,V%,TF%,TM%,PF%;OUT:poli$);

BEGIN
    indietro% := 1;
    PF_high% := INT( PF% / 4096 );
    PF_low% := INT( PF% - (PF_high% * 4096));

    IF ( verso% = indietro% )
    THEN BEGIN
            A8% := A8% OR 240;
            V% := V% OR 240;
            PF_high% := PF_high% OR 240;
    END;

    poli$ := CHR$(A8%) + CHR$(V%) + CHR$(TF%) + CHR$(TM%)
            + CHR$(PF_low%) + CHR$(PF_high%);

    OUTPUT: poli$;
END;
```

Ora vediamo la procedura di calcola_errori.
Occorre precisare che in realta´ ogni sistema di controllo reale non e´ mai di precisione assoluta.
Cosi´ accade anche per il pur sofisticato sistema di controllo dei motori, che, anche se impostato per garantire, per esempio, una velocita´ costante, non puo´ assicurarlo istantaneamente.
Sicuramente la velocita´ sara´ in media costante ma con piccole variazioni istantanee imprevedibili.
Cio´ implica che una traslazione puo´ non essere del tutto rettilinea e una rotazione puo´ anche causare spostamenti della BASE.

Conseguenza di questo e´ il fatto che noi alla fine del movimento sapremo la distanza percorsa e potremo calcolare l´orientamento finale della BASE, ma non potremo sapere la reale posizione della BASE con certezza.
La procedura calcola errori da quindi in uscita solo la distanza effettivamente percorsa e la eventale variazione di orientamento della BASE a movimento finito rispetto a quello che aveva alla partenza.

```
calcola_errori(IN:errore_dx%,errore_sx%;OUT:err_distanza,err_angolo;

BEGIN
    zero_approx := 0,47;
    alta% :=1;

    IF ( marcia% = alta% )
    THEN BEGIN
            conv_spazio := 17,6;
    END;

    ELSE BEGIN
            conv_spazio := 70,42;
    END;

    errore_dx% := errore_dx% / conv_spazio;
    errore_sx% := errore_sx% / conv_spazio;

    IF ( ABS( errore_dx% - errore_sx% ) < zero_approx )
    THEN BEGIN
            err_angolo := 0;
            err_distanza := errore_dx;
    END;

    ELSE BEGIN
            err_ang_rad = (errore_sx% - errore_dx%) / 360;

            IF ( ABS(errore_dx% + errore_sx%) < zero_approx )
            THEN BEGIN
                    err_distanza := 0;
            END;

            ELSE BEGIN
                    err_distanza := errore_sx -
                                    ((27/2) * err_ang_rad);
            END;
    err_angolo := ( 360/6,28) * err_ang_rad;

    END;

    OUTPUT: err_distanza,err_angolo;
END;
```

La procedura muovi_BASE richiede qualche conoscenza in piu´ sulla comunicazione con la BASE, per cui la descriviamo nel successivo paragrafo.

C.5.3: Come si colloquia con la BASE ?

Abbiamo visto che la BASE MODULUS si controlla inviandole una serie di PARAMETRI/POLINOMIO di movimento.
Abbiamo anche gia' menzionato la possibilita' di richiedere informazioni sullo STATO BASE (ad esempio motori accesi/spenti, urti subiti/non subiti, errore finale di spostamento, ...).
Questi sono messaggi del protocollo di comunicazione a livello applicativo: applicativo nel senso che il tipo di messaggi e le regole di colloquio sono legate all'applicazione della BASE MODULUS, cioe' al controllo del suo movimento.
Vedremo nel capitolo D cosa e' un protocollo di comunicazione e come realizzarlo, e in particolare nel paragrafo D.5 quali sono tutti i messaggi e le regole del livello applicativo del protocollo.
Per ora ci servono ancora due messaggi del livello applicativo; vediamo insieme da dove sorge questa esigenza.

Il microcalcolatore che controlla la BASE riserva, per ognuno dei due motori, un'area di memoria per accogliere i parametri/polinomio di movimento, ricevuti dal computer di controllo o dalla RF-CK, fintantoche' non siano stati eseguiti.
Nelle versioni di MODULUS piu' evolute, dotate di CPU a 16 bit a bordo, tali parametri arriveranno direttamente da tale CPU.

In questa area di memoria, chiamata BUFFER PARAMETRI o BUFFER di POLINOMIO, c'e' posto per due insiemi di parametri/polinomio: sara' cosi' possibile, per esempio, immagazzinare due successivi insiemi di parametri/polinomi per ogni motore.
Questo evita parte dei problemi dovuti al non trascurabile tempo necessario per inviare i parametri dal computer di controllo alla BASE.
Infatti si puo' inviare il successivo polinomio quando il primo e' in esecuzione: quando il primo e' stato esaurito viene eseguito il secondo (se raccordato) e si puo' nel frattempo inviare il terzo, e cosi' via.

Se c'e' spazio per uno o due polinomi nel buffer, questo si dice disponibile o libero (buffer free).
Se invece il buffer contiene gia' due polinomi si dice occupato (buffer busy).
A questo punto e' chiaro che per mandare un insieme di parametri/polinomio occorre sapere se il buffer del relativo motore e' libero.
Per questa ragione esiste un messaggio di "STATO BUFFER PARAMETRI" e anche un messaggio-comando con cui il computer di controllo puo' richiederlo.

Occorre poi specificare alla BASE di iniziare l'esecuzione del movimento descritto dai parametri che si trovano nel buffer parametri del relativo motore.
Occorre cioe' un comando-messaggio di "START MOTORE".

Vediamo come si puo´ svolgere il dialogo, a livello applicativo, fra MODULUS ed il computer di controllo:

1.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx disponibili,motori fermi,niente urti
2.COMPUTER: ecco polinomio-1 per motore sx
BASE MODULUS:ok,buffer dx e sx disponibili
3.COMPUTER: ecco polinomio-2 per motore sx
BASE MODULUS:ok,buffer sx occupato,buffer dx disponibile
4.COMPUTER: ecco polinomio-1 per motore dx
BASE MODULUS:ok,buffer sx occupato,dx disponibile
5.COMPUTER: ecco polinomio-2 per motore dx
BASE MODULUS:ok,buffer sx e dx occupati
6.COMPUTER: start motore dx e sx
BASE MODULUS:ok, buffer dx e sx occupati
7.COMPUTER: dammi stato buffer
BASE MODULUS:ok,buffer dx e sx occupati
8.COMPUTER: dammi stato buffer
BASE MODULUS:ok,buffer dx libero,sx occupato
9.COMPUTER: ecco polinomio-3 per motore dx
BASE MODULUS:ok,buffer dx e sx occupati
10.COMPUTER: dammi stato buffer
BASE MODULUS:ok,buffer dx occupato,sx libero
11.COMPUTER: ecco polinomio-3 per motore sx
BASE MODULUS:ok,buffer dx e sx occupati
12.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx occupati,motori accesi,niente urti
13.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx disponibili,motori accesi,niente urti
14.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx disponibili,motori fermi,urto avvenuto sul bumper 3,errore su posizione finale +32 uep a dx e -12 uep a sx
15.COMPUTER: ecco polinomio-1 per motore dx e sx
BASE MODULUS:ok,buffer dx e sx disponibili
16.COMPUTER: start motore dx e sx
BASE MODULUS:ok, buffer dx e sx disponibili
17.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx disponibili,motori accesi,niente urti
18.COMPUTER: dammi stato BASE
BASE MODULUS:buffer dx e sx disponibili,motori fermi,niente urti, errore su posizione finale +5 uep a dx e +5 uep a sx

Quello che abbiamo seguito qui e´ l´esecuzione di un movimento descritto da 3 polinomi raccordati, per entrambe i motori (sequenze da 1 a 13), sospeso da un urto sul bumper 3 (sequenza 14).
Sicuramente non era un movimento a tempi uguali per i due motori, visto che il buffer di polinomio del motore destro si libera prima del sinistro (sequenza 8).
Dopo l´urto si e´ comandata l´esecuzione di uno stesso polinomio per i due motori (quindi con moto rettilineo: sequenza da 15 a 17) e si e´ seguita l´esecuzione del movimento fino al suo termine (motori fermi: sequenza 18).

Vediamo cosa ci serve per realizzare la procedura muovi-BASE descritta nel paragrafo C.5.1: occorre inviare una serie di polinomi raccordati, far partire il movimento e seguirne lo svolgimento; a motori fermi si leggeranno gli errori di movimento nello stato BASE.

Siccome abbiamo scelto di usare solo traslazioni e rotazioni, i tempi di movimento TM saranno uguali per ogni polinomio destro e sinistro corrispondenti.
Siccome questo ci semplifica parecchio la vita nell´inviare i polinomi, costruiamo la procedura con questa ipotesi.

Per inviare i comandi-messaggi useremo delle procedure di protocollo a livello linea che troverete descritte nel paragrafo D.4 e D.7 .
Per ora accontentiamoci di sapere che sono le

```
inoltra_comando(IN:comando$;OUT:stato_buffer%)
inoltra_richiesta_stato(IN:-;OUT:stato_base$)
```

e servono, rispettivamente, l´una per inviare un comando_messaggio alla BASE (richiesta stato buffer, start motori, parametri/polinomio, ...) e l´altra per richiedere lo stato BASE.

Siccome la procedura muovi_base e´ complessa, diamo prima un flusso a livello descrittivo:

```
muovi_BASE(IN:poli_dx,poli_sx;OUT:urto,errore_dx,errore_sx)

BEGIN
    estrai_un_polinomio da poli_sx$ e poli_dx$;
    invia_polinomi_estratti;
    invia_comando start_motori;
    tempo_di_movimento := 0 secondi;
    WHILE (ci sono polinomi da inviare e non ci sono stati urti)
    DO BEGIN
           aspetta tempo_di_movimento;
           REPEAT
                richiedi stato_buffer;
           UNTIL ( stato_buffer = disponibile );
           controlla_urti_e_anomalie;
           estrai_un_polinomio da poli_sx$ e poli_dx$;
           invia_polinomi_estratti;
           calcola tempo_di_movimento;
    END;
    IF ( non c´e´ stato urto )
    THEN BEGIN
             REPEAT
                  chiedi stato_BASE;
             UNTIL ( motori fermi );
    END;
    chiedi errore_dx_e_sx;
    IF ( c´e´ stato urto )
    THEN BEGIN
             calcola_punto_fermata_movimento;
             calcola_vero errore_dx_e_sx;
    END;
    OUTPUT urto,err_dx,err_sx;
END;
```

```
muovi_BASE(IN:poli_dx$,poli_sx$;;OUT:urto%,errore_dx%,errore_sx%)

BEGIN
    start% := 80;
    polinomio% := 24;
    stato$ := CHR$(0);
    motore_sx% := 0;
    motore_dx% := 1;
    motori_dx&sx% := 3;
    free% := 0;
    spenti% := 0;

    un_pol_sx$ := LEFT$(poli_sx$,6);
    poli_sx$ := RIGHT$(poli_sx$,LEN(poli_sx$)-6);
    un_pol_dx$ := LEFT$(poli_dx$,6);
    poli_dx$ := RIGHT$(poli_dx$,LEN(poli_dx$)}6);

    IF ( un_pol_sx$ = un_pol_dx$ )
    THEN BEGIN
            op_code$ := CHR$(polinomio% OR motori_dx&sx%);
            messaggio$ := op_code$ + un_pol_sx$;
            inoltra_comando(IN:messaggio$ ;OUT:stato_buffer%)
    END;

    ELSE BEGIN
            op_code$ := CHR$(polinomio% OR motore_dx%);
            messaggio$ := op_code$ + un_pol_dx$;
            inoltra_comando(IN:messaggio$ ;OUT:stato_buffer%);
            op_code$ := CHR$(polinomio% OR motore_sx%);
            messaggio$ := op_code$ + un_pol_sx$;
            inoltra_comando(IN:messaggio$ ;OUT:stato_buffer%);
    END;

    comando$ := CHR$(start% OR motori_dx&sx%);
    inoltra_comando(IN:comando$ ;OUT:stato_buffer%);
    tempo_esecuz := 0;
    status% := 0;
    azzera_contasecondi;
    num_poli_esec% := 1;

    WHILE ( (len(poli_dx$)<>0) AND (urto%=0) )
    DO BEGIN
       REPEAT
            leggi_contasecondi(IN:- ;OUT:conta_secondi);
       UNTIL ( conta_secondi < tempo_esecuz );
       REPEAT
            inoltra_comando(IN:stato$ ;OUT:stato_buffer%)
            buffer% := stato_buffer% AND 48;
       UNTIL ( buffer% = free% );

       inoltra_richiesta_stato(IN:- ;OUT:stato_base$);
       urto% := ASC(MID$(stato_base$,8,1));
       status% :=ASC(MID$(stato_base$,9,1)) AND 3;

       IF ( (status% <> 0) AND (urto% = 0) )
       THEN BEGIN
```

```
                urto% := 255;
        END;

        un_pol_sx$ := LEFT$(poli_sx$,6);
        poli_sx$ := RIGHT$(poli_sx$,LEN(poli_sx$)_6);
        un_pol_dx$ := LEFT$(poli_dx$,6);
        poli_dx$ := RIGHT$(poli_dx$,LEN(poli_dx$)_6);

        IF ( un_pol_sx$ = un_pol_dx$ )
        THEN BEGIN
                op_code$ := CHR$(polinomio% OR motori_dx&sx%);
                messaggio$ := op_code$ + un_pol_sx$;
                inoltra_comando(IN:messaggio$
                                ;OUT:stato_buffer%);
        END;

        ELSE BEGIN
                op_code$ := CHR$(polinomio% OR motore_dx%);
                messaggio$ := op_code$ + un_pol_dx$;
                inoltra_comando(IN:messaggio$
                                ;OUT:stato_buffer%);
                op_code$ := CHR$(polinomio% OR motore_sx%);
                messaggio$ := op_code$ + un_pol_sx$;
                inoltra_comando(IN:messaggio$ ;OUT:stato_buffer%);
        END;

        num_poli_esec% := num_poli_esec% + 1;
        tempo_esecuz := ASC(MID$(un_pol_sx$,4,1) / 24;
        azzera_contasecondi;
END;

IF ( urto% = 0 )
THEN BEGIN
        REPEAT
                inoltra_richiesta_stato(IN:- ;OUT:stato_base$);
                urto% := ASC(MID$(stato_base$,8,1));
                status% :=ASC(MID$(stato_base$,9,1)) AND 3;
                motori% :=ASC(MID$(stato_base$,10,1)) AND 3;

                IF ( (status% <> 0) AND (urto% = 0) )
                THEN BEGIN
                        urto% := 255;
                END;
        UNTIL ( motori% = spenti% );
END;

err_sx_high% := ASC(MID$(stato_base$,1,1);
err_sx_low% := ASC(MID$(stato_base$,2,1);
err_dx_high% := ASC(MID$(stato_base$,3,1);
err_dx_low% := ASC(MID$(stato_base$,4,1);
err_sx% := (256 * err_sx_high%) + err_sx_low%;
err_dx% := (256 * err_dx_high%) + err_dx_low%;
err_sx% := err_sx% - 32768;
err_dx% := err_dx% - 32768;

IF ( urto% <> 0 )
```

```
    THEN BEGIN
            pol_err% :=ASC(MID$(stato_base$,9,1)) AND 8;

            IF ( pol_err% = 0 )
            THEN BEGIN
                    num_poli_esec% := num_poli_esec% - 1;
            END;

            calcola_errore_vero(IN:err_sx%,err_dx%,poli_sx$,poli_dx$,
                                num_poli_esec% ;OUT:err_sx%,err_dx%);

    END;
END;
```

Non commentiamo dettagliatamente questa procedura, perche´, per seguirlo, occorre prima che voi leggiate il capitolo D.
Inoltre questa e´ solo una proposta: il modo per realizzare il protocollo di comunicazione a livello applicativo dipende strettamente dalle esigenze e dalle scelte del progettista.

ATTENZIONE: la procedura funziona con il presupposto che i corrispondenti polinomi del motore destro e sinistro abbiano lo stesso tempo di movimento TM.
Se cosi´ non fosse occorre tenere presente separatamente l´evoluzione del movimento sinistro e destro.

Precisiamo poi che con "contasecondi" intendiamo la parte "secondi" dell´orologio/timer di sistema (orologio software o hardware) che ormai ogni personal o home computer possiede.
Nella traduzione in istruzioni per il vostro computer dovrete tenere conto delle specifiche modalita´ operative.
L´importante e´ avere le funzioni di "azzera_contasecondi" e di "leggi_contasecondi".
La parte che attende tempo e´ stata messa per alleggerire il "traffico" dati nel canale di comunicazione fra computer di controllo e BASE MODULUS: volendo si puo´ togliere.

La sottoprocedura calcola_errore_vero non la dettagliamo qui.
Occorrera´ calcolare lo spazio totale che si doveva percorrere con tutti i polinomi.
Poi calcolare lo spazio effettivamente percorso: si calcola lo spazio percorso con i polinomi eseguiti, che possiamo identificare conoscendo il numero dei polinomi eseguiti num_poli_esec, meno l´errore destro e sinistro rispettivamente.
L´errore vero sara´ la differenza fra spazio totale previsto e spazio effettivamente percorso.

PROCEDURA DI CALIBRAZIONE UNITA´ DI LUNGHEZZA

Nel paragrafo C.4 abbiamo parlato di procedura di calibrazione.
Ora abbiamo i mezzi per descriverla.

Prendiamo la BASE, selezioniamo la marcia bassa e poniamola ad una distanza D, misurata con precisione, da un ostacolo (per esempio un muro).
La distanza si misura dall´ostacolo al perno delle ruote della BASE, che corrisponde al punto di contatto delle ruote con il suolo.

Inviate un polinomio uguale per i due motori, che produca un movimento a velocita´ V per una distanza S, maggiore di D.

Date la partenza ai motori e aspettate l´urto.
Ad urto avvenuto chiedete lo stato BASE, leggete il campo errore_dx ed errore_sx.
Per ogni motore, il valore letto in UEP corrisponde alla distanza S - D (in centimetri), per cui:

1 UEP reale = (S - D) / errore letto [in centimetri]

ovvero, sapendo che 1 UEP teorico = 0,056 cm (marcia bassa) introduciamo una costante C di calibrazione delle costanti di conversione.

1 UEP reale = C * 1 UEP teorico

quindi C = 1 UEP reale / 1 UEP teorico

cioe´

C = (S - D) / (errore letto * 0,056)

Ricaveremo allora un C_destro e un C_sinistro che usemo nei calcoli di conversione fra cm e UEP.

TEMPO (in UTA) = 24 * TEMPO (in secondi)

marcia bassa: - - -

SPAZIO (in UEP) = 70,42 * C * SPAZIO (in cm)

VELOCITA´ (in UEP/UTA) = 2,93 * C * VELOCITA´ (in cm/sec)

ACCELERAZIONE (in UES/UTA^2) = 0,12 * C * ACCELERAZIONE (in cm/sec^2)

marcia alta: - - -

SPAZIO (in UEP) = 17,6 * C * SPAZIO (in cm)

VELOCITA´ (in UEP/UTA) = 0,73 * C * VELOCITA´ (in cm/sec)

ACCELERAZIONE (in UES/UTA^2) = 0,03 * C * ACCELERAZIONE (in cm/sec^2)

D ---------- COMUNICAZIONE E CONTROLLO ----------

D.1: Comunicazione: principi

Quando due corrispondenti devono comunicare fra loro occorre che si intendano correttamente.
Perche´ si intendano occorre stabilire delle regole di comunicazione comuni: chiamiamo PROTOCOLLO di comunicazione l´insieme delle regole (procedure) che regolano lo scambio di informazioni fra i corrispondenti.

Tanto per fare un esempio, pensiamo ad una telefonata fra due persone.
Bisogna sapere il numero del corrispondente: ci sono delle "regole" per trovarlo consultando l´elenco telefonico.
Bisogna eseguire la chiamata telefonica: c´e´ una precisa "regola" per effettuarla (alzare la cornetta, aspettare il segnale acustico, se libero comporre il numero, ...).
Una volta stabilita la connessione, ovvero una volta che il corrispondente ha sollevato la cornetta, occorre avere un linguaggio comune per intendersi: per esempio l´italiano (se si telefona alle Seychelles, per esempio, non e´ cosi´ scontato ...).
C´e´ una "regola" per rispondere (... pronto chi parla?... sono XXX ... non ho capito, puo´ ripetere? ... XXX...ho capito, cosa desidera?...).
C´e´ poi il contenuto della telefonata che, per essere inteso, ha di solito bisogno di seguire le regole del normale modo di pensare.
Insomma un sacco di convenzioni!

Osserviamo che la comunicazione avviene per livelli: c´e´ il mezzo trasmissivo (cavo, radio,...), il tipo di segnali che si inviano attraverso il mezzo, il modo con cui si usano i segnali per formare un messaggio e il loro significato, il modo per assicurarsi che il messaggio sia arrivato al corrispondente e in caso negativo ripeterlo, ...

E´ conveniente allora dividere le regole, cioe´ il protocollo, in livelli che rispecchino i vari aspetti della comunicazione.

Una impostazione sistematica del problema della comunicazione e´ stata effettuata dall´ ISO (Organizzazione Internazionale per gli Standard), introducendo 7 livelli di "intesa" e relative "regole".
Gli interessati possono consultare il paragrafo G.3 per sapere dove soddisfare le loro curiosita´.

Noi, piu´ semplicemente, useremo una suddivisione in 4 livelli di protocollo:

- 1. livello FISICO : comprende le specifiche fisiche della connessione: mezzi fisici, tipo di segnali trasmessi.
  A questo livello si parla di segnali elettrici, cavi/fili di connessione, connettori, tipi di interfaccia, velocita' di trasmissione, ...
  Tratteremo questo livello nei paragrafi D.2 e D.3 .

- 2. livello LINEA : comprende le specifiche di come i segnali descritti al livello 1 sono organizzati in simboli, i simboli in messaggi e definisce le regole per ottenere un efficace scambio di messaggi fra i corrispondenti.
  A questo livello si parla di codici, caratteri di controllo, blocchi dati, pacchetti,... . Spesso e' quel livello che si intende quando si parla semplicemente di "protocollo".
  Tratteremo questo livello nel paragrafo D.4

- 3. livello RETE : comprende le specifiche che regolano lo scambio dei messaggi quando esiste piu' di un solo possibile corrispondente e questi sono collegati tra loro con una rete.
  A questo livello si parla di punto a punto, multipunto, indirizzi di stazione,...
  Spesso i livelli 2 e 3 vengono descritti insieme, e cosi' faremo noi nel paragrafo D.4 .

- 4. livello APPLICAZIONE: riguarda lo scambio di messaggi fra i programmi applicativi che "girano" sulle macchine che stanno comunicando tra loro attraverso i livelli 1, 2 e 3.
  I tipi di messaggi e le regole sono strettamente connessi all'applicazione.
  Per esempio, nel nostro caso il programma che creerete sul vostro computer sara' interessato a gestire il movimento del robot: i messaggi riguarderanno il controllo del movimento e le regole saranno adeguate a questo.
  Tratteremo questo livello nel paragrafo D.5 (e ,in parte, anche nel D.6).

Il paragrafo D.7 da gli strumenti per "fare da se" sul proprio computer i protocolli ai vari livelli.
Per poterlo utilizzare occorre pero' aver letto prima i paragrafi precedenti.

BUONA LETTURA !

D.2: Livello fisico: collegamento via cavo

Questo paragrafo riguarda il livello fisico del protocollo di comunicazione con MODULUS nel caso la connessione avvenga via cavo. Tale connessione e' prevista solo con Commodore 64 o 128.

Vediamo subito perche'.
La connessione avviene usando l'interfaccia seriale "tipo RS 232" entrocontenuta nel Commodore 64 e 128. Diciamo "tipo RS 232" perche' Commodore usa livelli elettrici di interfaccia e connettori diversi da quelli previsti dallo standard RS 232 e non direttamente compatibili: per ottenere una interfaccia RS 232 compatibile occorre aggiungere un modem adapter.
Ecco perche' la connessione e' prevista solo per Commodore 64 e 128.
In ogni caso, livelli e connettore a parte, le specifiche funzionali dell'interfaccia RS 232 Commodore sono le stesse della RS 232 standard e quindi continueremo a chiamarla RS 232.

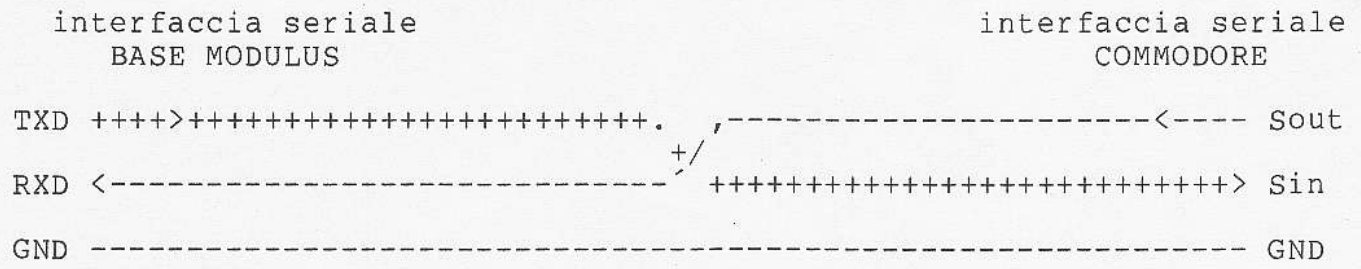
L'interfaccia RS 232 specifica i livelli elettrici dei segnali da usare, il tipo di connettore, una serie di fili per trasmettere e ricevere serialmente dei segnali digitali, e una lunga serie di fili di controllo modem.
Il MODEM e' un apparecchio che si incarica di trasmettere a lunga distanza dei segnali digitali usando segnali analogici.
Originariamente l'interfaccia RS 232 e' stata concepita per standardizzare il collegamento fra un terminale o un computer (chiamati DTE) ed un modem (chiamato DCE).
Successivamente interfacce di questo tipo sono state usate per realizzare direttamente collegamenti a distanza ridotta (max 15 metri) fra diverse apparecchiature.

Tutto cio' per spiegare che nel nostro caso non utilizzeremo modem e quindi nemmeno i segnali di controllo modem.
Tale modo di funzionamento viene normalmente chiamato "a tre fili" (three wire) o anche senza "handshake".

Non a caso si parla di 3 fili: si usano di fatto 3 fili.
Di questi tre fili, due vengono usati per convogliare i segnali da e per la BASE MODULUS e uno serve da riferimento comune per i livelli dei segnali.

La tensione elettrica e' sempre definita rispetto ad un livello di riferimento: ecco perche' insieme ai fili che convogliano i segnali elettrici occorre un filo che "trasporti" anche il livello di riferimento.

I tre fili o linee vengono comunemente denominati TXD (transmitted data = dati trasmessi), RXD (received data = dati ricevuti) e GND (ground = massa, terra).
Quando si realizza un collegamento direttamente sfruttando i fili dell´interfaccia RS 232 (senza modem) occorre "rovesciare i segnali" ovvero costruire un "cavo incrociato".



Cioe´ i dati trasmessi da un apparecchio diventano i dati ricevuti dall´altro e viceversa.
Per la realizzazione effettiva dei cavi e connettori consultare il paragrafo F.1 .
L´insieme dei 3 fili costituisce il canale di comunicazione.
Siccome su un filo viaggiano i segnali verso la BASE e sull´altro quelli verso il computer, la comunicazione puo´ avvenire contemporaneamente: questo tipo di collegamento viene di solito chiamato FULL-DUPLEX.

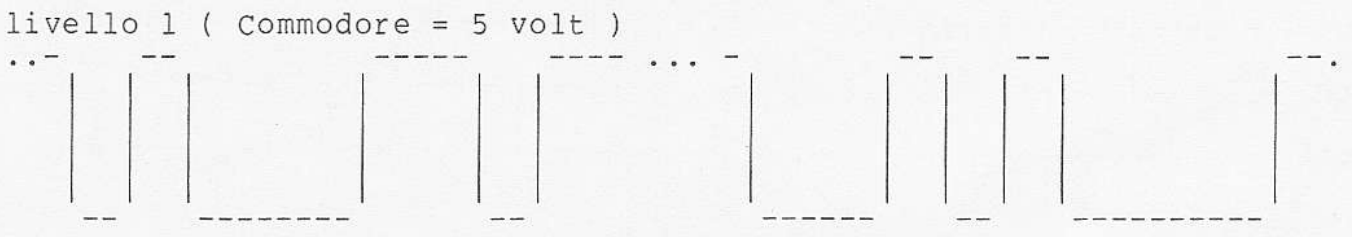
Vedremo pero´ che gli strati successivi di protocollo vieteranno l´uso in contemporanea dei due versi di comunicazione: logicamente il canale FULL-DUPLEX viene usato come fosse HALF-DUPLEX o TWO WAY ALTERNATE (= un verso alla volta).

I dati vengono trasmessi serialmente: cioe´ un bit dopo l´altro.
Visto che ogni bit puo´ assumere due valori, si usano due livelli elettrici (di tensione) : un livello corrispode a 0 e l´altro a 1.
Per l´interfaccia seriale Commodore il livello 1 corrisponde a circa 5 volt ed il livello 0 corrisponde a circa 0 volt.

Nel nostro canale inviamo un bit alla volta: questi pero´ vengono raggruppati in caratteri. Un carattere e´ un insieme di bit a cui e´ associato un significato preciso.
Il significato puo´ essere un simbolo: ad esempio la lettera A, il simbolo 1 o qualsiasi altro tasto della tastiera del vostro computer.
Spesso tali simboli vengono codificati con un insieme di caratteri di 7 bit chiamati ASCII.
Il significato puo´ anche essere un numero espresso in binario: per esempio se invio il carattere 00001010 intendo esattamente il numero binario 00001010 (10 decimale o A esadecimale).
Questo e´ il nostro caso: MODULUS usa caratteri binari di 8 bit.
Inviamo, in sostanza, un byte per volta: ogni carattere contiene quindi un numero compreso fra 0 e 255 (in decimale).

Significato a parte, ogni carattere deve poter essere distinto dal carattere precedente e dal successivo.
Un metodo usato e' quello di marcare l'inizio e la fine del carattere rispettivamente con uno START bit ed uno STOP bit: questo permette di lasciare trascorrere un tempo arbitrario fra due successivi caratteri.

Vediamo come viene inviato il numero (lo chiameremo poi byte, essendo il numero contenuto in un byte appunto) 141 seguito dal numero 10.
NB: 141 dec = 10110001 bin e 10 dec = 00001010 bin.



La linea e' a livello 1 a riposo, lo start bit e' un bit a 0 e lo stop bit e' un bit a 1.
Inoltre un segnale cosi' trasmesso si dice codificato NRZ (Non Ritorno a Zero) perche' due successivi bit a 1 vengono codificati tenendo la linea a livello 1 senza inserire ritorni a livello 0.
Vedremo che via radio si usera' una codifica diversa.

Per ogni carattere trasmesso e' possibile introdurre dei bit supplementari per rivelare eventuali errori di ricezione.
Il controllo piu' comune' e' quello di parita' che aggiunge un bit di parita' pari (o dispari) al carattere in modo che, in un byte, il numero totale dei bit a 1 sia pari (o dispari).
Questo sistema di controllo viene di solito usato quando si trasmettono caratteri con codifica ASCII.
Nel nostro caso non adottiamo controllo di parita': ci pensera' lo strato successivo (livello linea) di protocollo a rivelare gli errori.

Il tempo in cui la linea viene tenuta a livello 1 (o 0) per trasmettere un bit e' legato alla velocita' di trasmissione.
La velocita' di trasmissione si esprime infatti in bit al secondo (o baud - che coincide con bit/sec quando la codifica e' a due livelli, come in questo caso).

Se si trasmette a 300 baud o bit/sec, come succede nel nostro collegamento via cavo, significa che si possono trasmettere al piu´ 300 bit in un secondo ovvero che un bit dura 1/300 di secondo.
Siccome per noi un carattere e´ lungo 8 + 2 bit, la massima frequenza di trasmissione sara´ di 30 caratteri (o byte) al secondo.
Diciamo massima perche´, mentre il tempo di bit ha una durata definita (= 1/300 di secondo a 300 baud o piu´ in generale 1/(N baud) di secondo , se la velocita´ e´ di N baud), l´intervallo fra un carattere e il successivo e´ arbitrario e dipende dalle apparecchiature che dialogano e, in particolare, dai programmi che le gestiscono.
Quest´ultima caratteristica qualifica il tipo di trasmissione che viene cosi´ chiamato ASINCRONO. In una trasmissione asincrona solo l´intervallo fra due successivi bit di un carattere e´ sempre lo stesso, mentre il tempo fra due successivi caratteri e´ arbitrario.

Per esempio, se usate il normale BASIC del vostro computer per realizzare il protocollo di comunicazione con la BASE, lo stesso interprete di linguaggio potra´ introdurre ritardi variabili fra un carattere ed il successivo nella trasmissione.
Anche per questo le interfacce seriali dei personal computer adottano quasi esclusivamente il modo asincrono.

Riassumiamo: i parametri che dovrete programmare sull´interfaccia seriale del vostro Commodore sono:

- 8 bit di dati

- 1 bit di stop (lo start bit e´ sempre uno)

- niente controllo di parita´ (no parity)

- interfaccia a 3 fili (nessun handshake con segnali modem DTR,DSR,RTS,...)

- velocita´ 300 baud

- codifica binaria (non ASCII)

In BASIC Commodore la cosa si realizza cosi´:

```
10 SYS (49407):REM permette ricezione carattere 00000000 bin

20 OPEN n,2,0,CHR$(6):REM programma i/f seriale assegnandogli
                        file numero n
30 GET#n,A$:GET#n,A$:GET#n,A$:REM "pulizia" buffer ingresso RS 232
```

Occorre solo che decidiate il numero n di file da assegnare.
La routine in linguaggio macchina chiamata con SYS (49407) ha lo scopo di permettere la ricezione del carattere formato da tutti 0, che normalmente il Commodore "filtra".

La si introduce con il seguente programma BASIC:

```
4010 FOR I=0 TO 56:READ A:POKE 49152+I-1,A:NEXT
4020 SYS 49159
4030 FOR I=0 TO 17
4040 POKE 49412+I,PEEK(61574+I):NEXT
4050 FOR I=0 TO 6
4060 POKE 49444+I,PEEK(61569+I):NEXT
4070 FOR I=0 TO 9
4080 POKE 49454+I,PEEK(46221+I):NEXT
4090 FOR I=0 TO 14
4100 POKE 49224+I,PEEK(46231+I):NEXT
4110 FOR I=49407 TO 49411
4120 READ A:POKE I,A:NEXT
4130 FOR I=49430 TO 49443
4140 READ A:POKE I,A:NEXT
4150 FOR I=49451 TO 49453
4160 READ A:POKE I,A:NEXT
4170 FOR I=49464 TO 49482
4180 READ A:POKE I,A:NEXT
4190 FOR I=0 TO 2
4200 READ A:POKE 61574+I,A:READ A:POKE 46221+I,A:
     READ A:POKE 46244+I,A:NEXT
4210 FOR I=0 TO 1
4220 READ A:POKE 49421+I,A:NEXT
4230 FOR I=0 TO 1
4240 READ A:POKE 46227+I,A:NEXT
4250 FOR I=0 TO 1
4260 READ A:POKE 46239+I,A:NEXT
4270 FOR I=0 TO 2
4280 READ A:POKE 46241+I,A:NEXT
4290 SYS 49407
4300 DATA165,254,069,002,133,002,096
4310 DATA169,000
4320 DATA133,251,169,224,133,252,160,000
4330 DATA177,251,145,251,200,208,249,230
4340 DATA252,165,252,201,000,208,241,169
4350 DATA160,133,252,169,000,133,251,168
4360 DATA177,251,145,251,200,208,249,230
4370 DATA252,165,252,201,192,208,241,096
4380 DATA169,005,133,001,096,208,006,169
4390 DATA021,133,106,169,000,238,156,002
4400 DATA076,155,240,076,163,240,076,144
4410 DATA180,165,106,201,021,240,003,076
4420 DATA168,180,169,000,133,106,076,144
4430 DATA180,076,076,076,004,046,059,193
4440 DATA193,193,240,021,240,015,240,008
4450 DATA076,168,180
```

D.3: Livello fisico: collegamento via R.F. COMMAND KEYBOARD

Questo paragrafo riguarda il livello fisico del protocollo di comunicazione con MODULUS nel caso la connessione avvenga via radio attraverso la RF- COMMAND KEYBOARD.

In ogni caso, la connessione con il computer di controllo avviene attraverso una normale interfaccia seriale di cui la RF-CK e' dotata. L'interfaccia seriale della RF-CK accetta sia i livelli RS 232 standard che i livelli "tipo Commodore" per cui la connessione e' fattibile con qualsiasi computer dotato di interfaccia RS 232 standard e anche con Commodore 64 o 128.

Ricordiamo che l'interfaccia "tipo RS 232" di cui sono dotati i Commodore 64 e 128 usa livelli elettrici e connettori diversi da quelli previsti dallo standard RS 232 e non direttamente compatibili: per ottenere una interfaccia RS 232 compatibile occorre aggiungere un modem adapter.
Ecco perche' sono previste due connessioni divese: una per Commodore 64 e 128 e l'altra per normale RS 232.

Livelli e connettore a parte, le specifiche funzionali dell'interfaccia RS 232 Commodore sono le stesse della RS 232 standard.
Fatta questa precisazione continueremo a chiamare RS 232 anche l'interfaccia Commodore.

Abbiamo un doppio interfacciamento dunque: da una parte quello fra computer di controllo e RF-CK e dall'altra quello fra RF-CK e BASE MODULUS.
Dividiamo allora in due l'argomento.
Tenete presente che all'utente, cioe' a voi, il livello canale radio (normalmente chiamato RF = Radio Frequency) e' totalmente trasparente: avrete a che fare solo con il livello RS 232 dell'RF-CK.
Ragion per cui, se non siete interessati, potete tranquillamente saltare la descrizione del livello 1/RF.

Livello 1/RF

La RF-CK si comporta a tutti gli effetti come un modem, cioe´ un apparecchio che si incarica di trasmettere a lunga distanza dei segnali digitali usando segnali analogici.
Vedremo infatti che saremo costretti ad usare nel livello 1/RS 232 anche dei segnali detti di controllo modem (o di handshake).

Il canale di comunicazione e´ costituito da un canale a radiofrequenza.
Sia la RF-CK che la BASE MODULUS sono dotate di rice-trasmettitori di potenza consentita dalla legge.

Quando si usa MODULUS in configurazione piu´ evoluta, con la TECHNO-CAKE o spicchioteca, il ricetrasmettitore non e´ piu´ nella BASE ma viene ospitato nel "sistema nervoso" della spicchioteca stessa (il cluster controller che gestisce la rete di microprocessori di MODULUS dedicati alle varie funzioni) e sara´ in grado di trasmettere dati a velocita´ 4 volte piu´ elevata.

Il canale e´ usufruibile solo in una direzione per volta: da BASE verso RF-CK o viceversa.
La comunicazione non puo´ avvenire contemporaneamente: questo tipo di collegamento viene di solito chiamato HALF-DUPLEX o TWO WAY ALTERNATE (= un verso alla volta).

La modulazione usata per trasmettere i dati e´ la modulazione in ampiezza.
Se volete saperne di piu´ consultate i paragrafi G.1 e G.3, che vi suggeriranno cosa leggere.

I dati vengono trasmessi serialmente: cioe´ un bit dopo l´altro.
Si usano due livelli elettrici (di tensione) : un livello lo chiamiamo 0 e l´altro 1.

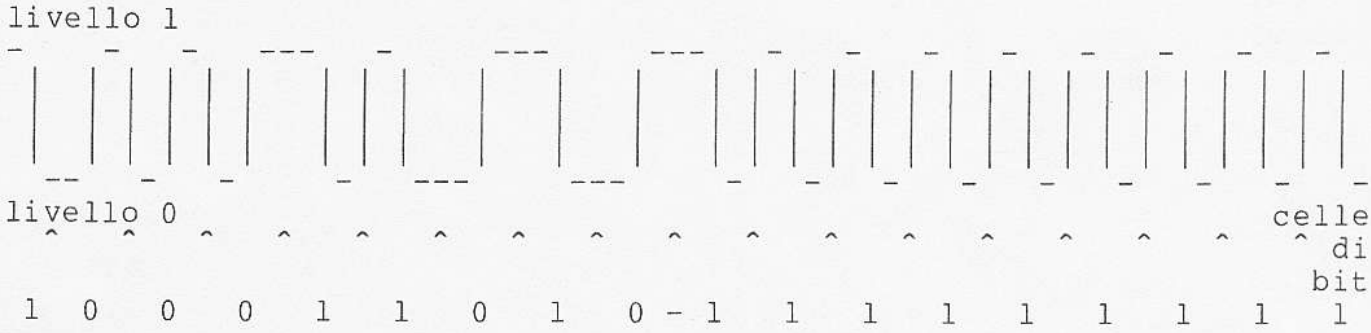
Nel nostro canale inviamo un bit alla volta: questi pero´ vengono raggruppati in caratteri. Un carattere e´ un insieme di bit a cui e´ associato un significato preciso.
Il significato puo´ essere un simbolo: ad esempio la lettera A, il simbolo 1 o qualsiasi altro tasto della tastiera del vostro computer.
Spesso tali simboli vengono codificati con un insieme di caratteri di 7 bit chiamati ASCII.
Il significato puo´ anche essere un numero espresso in binario: per esempio se invio il carattere 00001010 intendo esattamente il numero binario 00001010 (10 decimale o A esadecimale).
Questo e´ il nostro caso: MODULUS usa caratteri binari di 8 bit.
Inviamo, in sostanza, un byte per volta: ogni carattere contiene quindi un numero compreso fra 0 e 255 (in decimale).

I caratteri che la RF-CK o la BASE MODULUS trasmettono sono gia´ organizzati in messaggi, cioe´ "pacchetti" di byte (lo vedrete nei par D.4 e D.5): il tempo fra due byte successivi del pacchetto e´ costante.

Ogni messaggio viene trasmesso sul canale RF preceduto da una sequenza di preavviso/sincronizzazione composta da 16 bit alternativamente a 0 e 1 che servono a dare il tempo al trasmettitore RF di generare un segnale stabile, seguita da una configurazione di sincronismo di inizio blocco formata da un bit e mezzo a livello 1 e un bit e mezzo a livello 0.
Segue il pacchetto in cui ad ogni byte viene aggiunto un nono bit che e´ normalmente a 0.

Il nono bit e´ a 1 solo nel primo byte del messaggio e nell´ultimo, che e´ sempre un byte di fine pacchetto contenente tutti 1 (11111111 bin o FF hex).
Viene inoltre trasmesso, nel mezzo byte (nibble) piu´ significativo del primo byte, il numero progressivo del pacchetto, con conteggio modulo 8.

Ogni bit del pacchetto viene trasmesso con codifica tipo Manchester: un bit a 1 viene codificato con una transizione 1 --> 0, mentre un bit a 0 con una transizione 0 --> 1.
Vediamo come viene inviato il numero (lo chiameremo poi byte, essendo il numero contenuto in un byte appunto) 141 , che si suppone non il primo, seguito dal carattere di fine messaggio 255 (FF hex con nono bit a 1).
NB: 141 dec = 10110001 bin e 255 dec = 11111111 bin.



Per ogni carattere trasmesso e´ possibile introdurre dei bit supplementari per rivelare eventuali errori di ricezione.
Il controllo piu´ comune´ e´ quello di parita´ che aggiunge un bit di parita´ pari (o dispari) al carattere in modo che il numero totale dei bit a 1 sia pari (o dispari).
Questo sistema di controllo viene di solito usato quando si trasmettono caratteri con codifica ASCII.
Nel nostro caso non adottiamo controllo di parita´: ci pensera´ lo strato successivo (livello linea) di protocollo a rivelare gli errori.

Il tempo in cui la linea viene tenuta a livello 1 (o 0) per trasmettere un bit e´ legato alla velocita´ di trasmissione.

La velocita´ di trasmissione si esprime infatti in bit al secondo (o baud - che coincide con bit/sec quando la codifica e´ a due livelli, come in questo caso).
Se si trasmette a 300 baud o bit/sec, come succede nel nostro collegamento via cavo, significa che si possono trasmettere al piu´ 300 bit in un secondo ovvero che un bit dura 1/300 di secondo.

Siccome per noi un carattere e´ lungo 8 + 1 bit, la frequenza di trasmissione sara´ di 33 caratteri (o byte) al secondo, trascurando la sequenza di sincronismo iniziale.

In questo caso, il modo di trasmissione si dice SINCRONO, in quanto l´intervallo di tempo fra due successivi caratteri e´ costante.

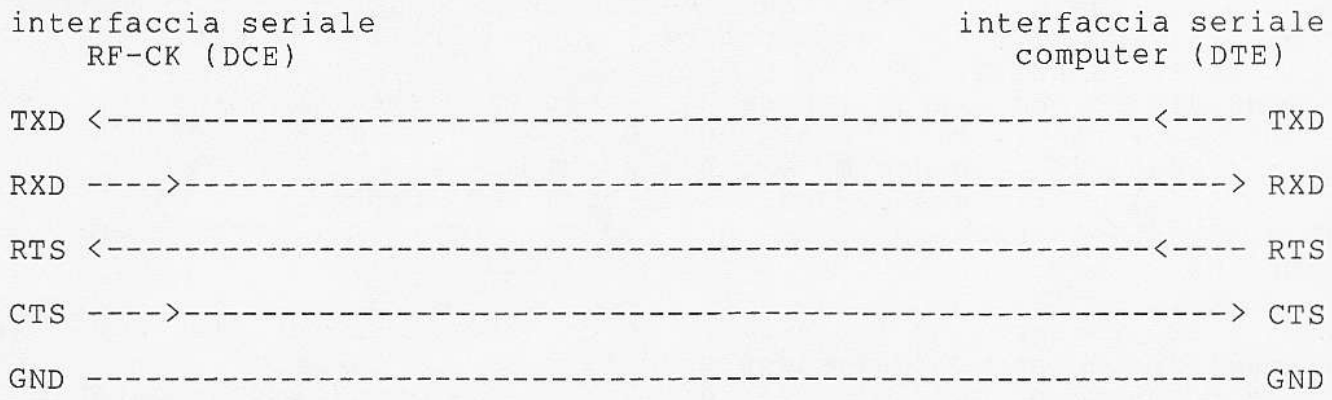
Quando si lavora con MODULUS in configurazione piu´ evoluta, che non la sola BASE, il canale seriale in RF opera alla velocita´ di 1200 bit/sec (o baud). Sulla RF-CK e´ presente quindi la possibilita´ di programmare tale velocita´ (vedi paragrafo A.4.3).

Livello 1/RS 232

L´interfaccia RS 232 specifica i livelli elettrici dei segnali da usare, il tipo di connettore, una serie di fili per trasmettere e ricevere serialmente dei segnali digitali, e una lunga serie di fili di controllo modem.

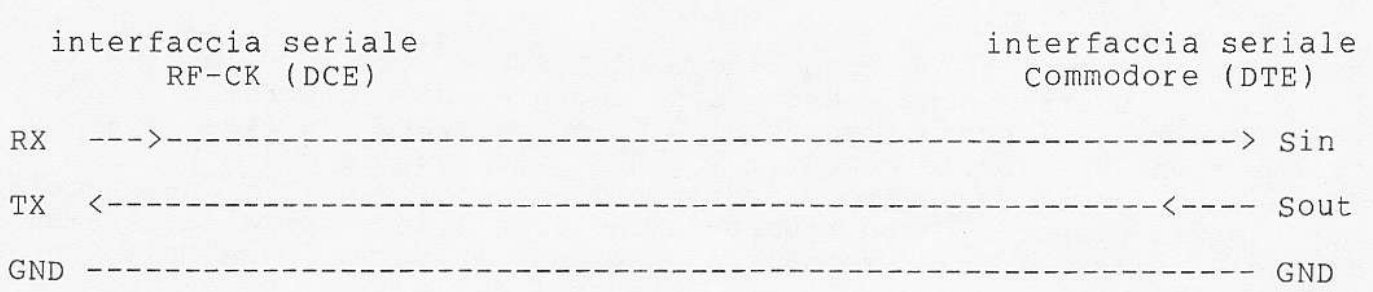
Il modem (o DCE) nel nostro caso e´ l´RF-CK (vedi livello 1/RF).
Occorrera´ usare anche parte dei segnali di controllo modem: sostanzialmente i segnali servono a sincronizzare la velocita´ di trasmissione del vostro computer (o DTE) con quella della RF-CK.
Se la RF-CK ha ricevuto troppi caratteri (byte) dall´ interfaccia seriale del computer e non riesce ad inoltrarli in tempo alla BASE sul canale RF allora l´RF-CK segnalera´ su un filo la sua momentanea indisponibilita´ a ricevere altri caratteri dalla linea RS 232.
Viceversa se l´interfaccia seriale del computer non "ce la fa a stare dietro" al ritmo con cui l´RF-CK gli invia i caratteri ricevuti via RF dalla BASE segnalera´ su un altro filo la sua indisponibilita´.
Tale modo di funzionamento viene normalmente chiamato "handshake".
Per l´interfaccia RS 232 del vostro computer il filo su cui segnalare l´indisponibilita´ si chiama RTS (Request To Send = richiesta di trasmettere), mentre il filo con cui ascolta la disponibilita´ (o meno) dell´RF-CK si chiama CTS (Clear To Send = pronto a trasmettere).
Nell´RF-CK, essendo questo un DCE, il significato dei segnali e´, per convenzione, l´inverso che per un DTE, per cui basta collegare RTS della RS 232 al RTS dell´RF-CK e idem per il CTS.

Ci sono poi i due fili o linee che convogliano i dati seriali dal computer verso la RF-CK e viceversa e vengono denominati TXD (transmitted data = dati trasmessi), RXD (received data = dati ricevuti) e il riferimento comune di massa GND (ground = massa, terra): anche per questi, come per RTS e CTS, e´ sufficiente collegare insieme i segnali del DTe e del DCE con lo stesso nome.



Per la realizzazione effettiva dei cavi e connettori consultare il paragrafo F.1 .

ATTENZIONE: per problemi di differenza di livelli di tensione elettrica, chi usa Commodore deve invece effettuare la seguente connessione (NB i segnali RX e TX sono diversi da RXD e TXD usati prima con la RS 232):



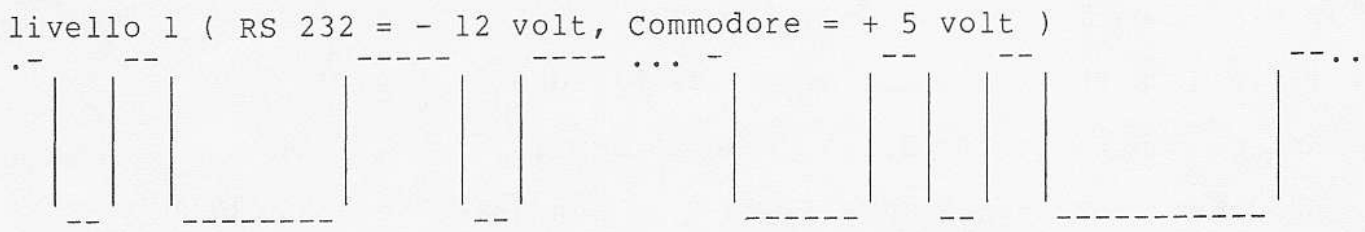
La comunicazione non puo´ avvenire contemporaneamente nei due sensi: questo dipende dalla struttura del canale RF.
Questo tipo di collegamento viene di solito chiamato HALF-DUPLEX o TWO WAY ALTERNATE (= un verso alla volta).

I dati vengono trasmessi serialmente: cioe´ un bit dopo l´altro.
Visto che ogni bit puo´ assumere due valori, si usano due livelli elettrici (di tensione) : un livello corrispode a 0 e l´altro a 1.
Per l´interfaccia seriale RS 232 il livello 1 corrisponde a circa -12 volt (Commodore + 5 volt) ed il livello 0 corrisponde a circa +12 volt (Commodore 0 volt).
Se vogliamo essere piu´ precisi, lo standard RS 232 prevede per il livello 1 (chiamato MARK) una tensione compresa fra -3 e -15 volt e per il livello 0 (chiamato SPACE) una tensione compresa fra +3 e +15 volt.

Nel nostro canale inviamo un bit alla volta: questi vengono pero´ raggruppati in caratteri. Un carattere e´ un insieme di bit a cui e´ associato un significato preciso.
Il significato puo´ essere un simbolo: ad esempio la lettera A, il simbolo 1 o qualsiasi altro tasto della tastiera del vostro computer.
Spesso tali simboli vengono codificati con un insieme di caratteri di 7 bit chiamati ASCII.
Il significato puo´ anche essere un numero espresso in binario: per esempio se invio il carattere 00001010 intendo esattamente il numero binario 00001010 (10 decimale o A esadecimale).
Questo e´ il nostro caso: MODULUS usa caratteri binari di 8 bit.
Inviamo, in sostanza, un byte per volta: ogni carattere contiene quindi un numero compreso fra 0 e 255 (in decimale).

Significato a parte, ogni carattere deve poter essere distinto dal carattere precedente e dal successivo.
Un metodo usato e´ quello di marcare l´inizio e la fine del carattere rispettivamente con uno START bit ed uno STOP bit: questo permette di lasciare un intervallo di tempo arbitrario fra due successivi caratteri.
Vediamo come viene inviato il numero (lo chiameremo poi byte, essendo il numero contenuto in un byte appunto) 141 seguito dal numero 10.

NB: 141 dec = 10110001 bin e 10 dec = 00001010 bin.



start 1 0 0 0 1 1 0 1 stop start 0 1 0 1 0 0 0 0 stop

La linea e´ a livello 1 a riposo, lo start bit e´ un bit a 0 e lo stop bit e´ un bit a 1.
Inoltre un segnale cosi´ trasmesso si dice codificato NRZ (Non Ritorno a Zero) perche´ due successivi bit a 1 vengono codificati tenendo la linea a livellol senza inserire ritorni a livello 0.
Vedremo che via radio si usera´ una codifica diversa.

Per ogni carattere trasmesso e´ possibile introdurre dei bit supplementari per rivelare eventuali errori di ricezione.
Il controllo piu´ comune´ e´ quello di parita´ che aggiunge un bit di parita´ pari (o dispari) al carattere in modo che il numero totale dei bit a 1 sia pari (o dispari).
uesto sistema di controllo viene di solito usato quando si trasmettono caratteri con codifica ASCII.
Nel nostro caso non adottiamo controllo di parita´: ci pensera´ lo strato successivo (livello linea) di protocollo a rivelare gli errori.

L´intervallo di tempo fra un carattere ed il successivo e´ arbitrario: questo modo di trasmissione viene detto ASINCRONO.
La velocita´ di trasmissione (o meglio baud rate) puo´ essere di 300 o 9600 baud. Questa corrisponde rispettivamente ad un massimo di 30 e 960 byte/secondo (per sapere perche´ vedi paragrafo D.2).
Quando si opera con Commodore occorre per forza selezionare 300 baud (perche´ 9600 baud non e´ consentita dall´interfaccia Commodore).

La velocita´ si seleziona sia sull´RF-CK che sul vostro computer. Per la selezione sulla RF-CK vedi paragrafo A.4.3 .

ATTENZIONE: se usate 300 baud allora il meccanismo di handshake diventa di fatto inutile ed e´ possibile evitare di collegare i fili RTS e CTS fra computer e RF-CK.
Questo e´ anche il caso Commodore, visto che puo´ usare solamente 300 baud.
Per la realizzazione dei cavi consultate il par F.1 .

Riassumendo: i parametri che dovrete programmare sull´interfaccia seriale del vostro computer sono:

- 8 bit di dati
- 1 bit di stop (lo start bit e´ sempre uno)
- niente controllo di parita´ (no parity)
- handshake con segnali RTS e CTS (no handshake se a 300 baud)
- velocita´ 300 o 9600 baud (300 se Commodore)
- codifica binaria (non ASCII)

In BASIC Commodore la cosa si realizza cosi´:

```
10 SYS (49407):REM permette ricezione carattere 00000000 bin
20 OPEN n,2,0,CHR$(6):REM programma i/f seriale assegnandogli
                         file numero n
30 GET#n,A$:GET#n,A$:GET#n,A$:REM "pulizia" buffer ingresso RS 232
```

Occorre solo che decidiate il numero n di file da assegnare.
La routine in linguaggio macchina chiamata con SYS (49407) ha lo scopo di permettere la ricezione del carattere formato da tutti 0, che normalmente il Commodore "filtra". La si introduce con il programma BASIC scritto in fondo al paragrafo D.2 (pag 7).

In BASIC per PC IBM e compatibili (a 300 baud):

```
10 OPEN "COM1:300,N,8,1,CS10,CD0,DS0,BIN"
```

o a 9600 baud:

```
10 OPEN "COM1:9600,N,8,1,CS10,CD0,DS0,BIN"
```

D.4: Livello logico: protocollo di linea

Il livello logico o di linea del protocollo e´ quello che si incarica di presiedere al buon invio di un blocco di dati, o messaggio, fra i due corrispondenti.
Questo livello include le specifiche della forma in cui il nostro messaggio verra´ inviato, normalmente chiamata pacchetto, le tecniche di rilevazione degli errori di ricezione e la loro correzione.

E´ importante sottolineare che a questo livello non importa il contenuto ed il significato del messaggio trasmesso o ricevuto, che invece riguarda il livello successivo (applicativo: MODULUS).
Ci si preoccupa soltanto del corretto inoltro del messaggio.

NOTA

In realta´ MODULUS e´ progettato come una architettura multi-microprocessore in cui i microprocessori sono interconnessi fra loro e al computer/CPU di controllo attraverso una rete di comunicazione.

Occorrera´ quindi anche un livello di protocollo di rete per la corretta gestione dello scambio messaggi.

In questo paragrafo e´ compresa anche la descrizione dello strato di protocollo di rete anche se usando MODULUS in configurazione sola BASE la rete e´ in realta´ formata solo dal computer di di controllo e dalla BASE stessa.

La struttura dei pacchetti scambiati dal protocollo di linea e´ leggermente diversa a seconda che si utilizzi la connessione via cavo (con Commodore) o via radio (con Commodore o altro computer).
Vediamola.

1. CONNESSIONE VIA CAVO: esistono due tipi di pacchetti: quello inviato da CBM (Commodore) verso MODULUS e quello inviato da MODULUS verso CBM.

CBM ---> MODULUS

IDENTIFICATORE SLAVE
MESSAGGIO LIVELLO "MODULUS"
CODICE CONTROLLO ERRORI
CODICE FINE PACCHETTO

MODULUS ---> CBM

IDENTIFICATORE SLAVE
NUMERO CARATTERI (BYTE) MESSAGGIO LIVELLO "MODULUS"
MESSAGGIO LIVELLO "MODULUS"
CODICE CONTROLLO ERRORI

2. CONNESSIONE VIA RADIO: anche qui esistono due tipi di pacchetti: quello inviato da Personal Computer verso MODULUS (via RF-CK) e quello inviato da MODULUS (via RF-CK) verso Personal Computer.

PC ---> MODULUS

NUMERO CARATTERI (BYTE) DI (MESSAGGIO LIVELLO "MODULUS" + IDENTIFICATORE SLAVE)
IDENTIFICATORE SLAVE
MESSAGGIO LIVELLO "MODULUS"
CODICE CONTROLLO ERRORI

MODULUS ---> PC

IDENTIFICATORE SLAVE
NUMERO CARATTERI (BYTE) MESSAGGIO LIVELLO "MODULUS"
MESSAGGIO LIVELLO "MODULUS"
CODICE CONTROLLO ERRORI

- IDENTIFICATORE SLAVE: questo campo e´ l´unico che riguarda il livello di rete del protocollo.
L´architettura di MODULUS e´ costituita da una rete di microprocessori dedicati alle varie funzioni (spostamento, voce, movimento braccia, mani, ...).
Chi coordina e comanda questi microprocessori e´ il computer/CPU di controllo, che per questo viene chiamato di solito MASTER (capo o padrone).
Ogni microprocessore dedicato, che esegue le funzioni richieste dal MASTER, viene chiamato SLAVE (schiavo) e viene identificato con un numero da 0 a 15.
Il protocollo di rete e´ quindi di tipo MASTER-SLAVE : c´e´ un computer "capo" e tanti computer "sudditi".
Il campo IDENTIFICATORE SLAVE identifica dunque il microprocessore a cui e´ diretto (o da cui proviene) il pacchetto.
Nella configurazione di MODULUS che possedete, la BASE e´ uno di questi SLAVE ed il computer di controllo e´ il vostro computer.
L´identificatore della BASE e´ 0.
Se l´identificatore slave e´ un numero superiore a 15, e precisamente compreso fra 48 e 83 (30 e 53 hex), allora il messaggio proviene direttamente dal microprocessore della RF-COMMAND KEYBOARD.
In questo caso il formato del pacchetto e´ diverso da quelli visti nella pagina precedente: consultare il paragrafo D.6 .

- CODICE CONTROLLO ERRORE : questo campo contiene un byte: il suo valore servira´ a chi riceve il pacchetto per verificare se il pacchetto si e´ "rovinato" per strada (causa interferenze o rumore in genere, per esempio).
Il tipo di algoritmo di rilevazione di errore adottato e´ quello del LONGITUDINAL REDUNDANCY CHECK a 8 bit: il campo di controllo viene percio´ chiamato brevemente LRC.
Il valore contenuto nel campo LRC viene generato da chi emette il pacchetto facendo l´OR esclusivo (XOR) fra tutti i byte del pacchetto precedenti il campo LRC stesso.
Chi riceve il pacchetto ricalcola l´LRC sul pacchetto ricevuto e confronta il valore ottenuto con quello ricevuto nel campo LRC.
Se sono uguali allora il pacchetto e´ integro.
Se sono diversi qualche bit e´ stato alterato "per strada" ed il pacchetto e´ da scartare.
Chi volesse saperne di piu´ segua i consigli del paragrafo G.1 .

- CODICE DI FINE MESSAGGIO : questo campo e' costituito da un byte contenente il valore 255 decimale (FF hex o 11111111 bin) che segnala la fine del pacchetto.

- NUMERO CARATTERI DI (MESSAGGIO + IDENTIF. SLAVE) : e' un campo di un byte che contiene il numero di caratteri (byte) che compongono il messaggio livello "MODULUS" piu' il numero di caratteri che compone il campo identificatore slave (cioe' 1).
  Questo campo deve contenere sempre un numero maggiore di zero.
  Viceversa si ricevera' in risposta un messaggio di "nack" (vedi a pie' di pagina).

- NUMERO CARATTERI DI MESSAGGIO LIVELLO "MODULUS" : e' un campo di un byte che contiene il numero di caratteri, o byte, che formano il messaggio a livello "MODULUS".
  Vedremo nel paragrafo D.5 che tale numero sara' solo 1 o 10.

- MESSAGGIO LIVELLO "MODULUS" : sono i messaggi ammessi dallo strato superiore di protocollo: quello applicativo MODULUS (vedi paragrafo D.5).

Visto il formato dei pacchetti vediamo ora le regole di protocollo.
Essendo il protocollo di tipo MASTER-SLAVE l'iniziativa e' sempre del MASTER, cioe' del computer di controllo.
Lo slave indirizzato (la BASE, nel nostro caso) risponde con un pacchetto "risposta" che conferma (o meno) la buona ricezione del pacchetto.
Il pacchetto che conferma la buona ricezione viene spesso chiamato "ack" (dall'inglese to acknowledge = riconoscere, accettare).
Il pacchetto di non conferma viene comunemente chiamato "nack" (not acknowledge).
Questi messaggi si trovano, per questioni di economia di trasmissione, in un mezzo byte (nibble) dell'unico byte che forma il campo messaggio livello "MODULUS" del pacchetto "risposta".
Di fatto mezzo byte (quello che contiene "ack" o "nack") appartiene al protocollo livello LINEA e l'altro mezzo byte appartiene al protocollo livello MODULUS.
Un valore di 6 decimale (0110 bin) nel nibble basso significa "ack".
Un valore di 9 decimale (1001 bin) nel nibble basso significa "nack".

Il nibble alto contiene informazioni sullo stato di occupazione dei buffer di polinomio dei motori destro e sinistro della BASE ed e´ dunque un messaggio a livello applicativo MODULUS (vedi paragrafo D.5).

Quindi per estrarre il livello LINEA occorrera´ mascherare il byte con 00001111 bin (cioe´ fare un AND con 15 dec).
Viceversa, per estrarre il livello MODULUS occorrera´ mascherare il byte con 11110000 bin (cioe´ fare un AND con 240 dec).

Il flusso da seguire per condurre il protocollo di linea con il vostro computer e´ questo (la notazione usa strutture di controllo pseudo PASCAL - vedi paragrafo E.1 e G.1 -).

```
BEGIN
   REPEAT
       trasmetti_un_pacchetto;

       ricevi_un_pacchetto;

       estrai_messaggio_modulus;

   UNTIL ( messaggio_modulus contiene "ack" );
END;
```

Cioe´ il computer di controllo trasmettera´ il pacchetto finche´ la BASE MODULUS non rispondera´ con un pacchetto contenente "ack".
La descrizione dei programmi che realizzano le funzioni TRASMETTI_UN_PACCHETTO, ecc. sono nel paragrafo D.7 .

ATTENZIONE: c´e´ pero´ un eccezione a questo flusso.
Esiste un pacchetto che contiene un particolare tipo di messaggio livello "MODULUS" (RICHIESTA STATO BASE, come vedremo nel paragrafo D.5) a cui la BASE MODULUS risponde con un pacchetto che non contiene il messaggio di conferma.
In tal caso conviene operare cosi´:

```
BEGIN
   REPEAT
       trasmetti_un_pacchetto ( contenente
                                RICHIESTA STATO );

       ricevi_un_pacchetto;

       estrai_messaggio_modulus;

   UNTIL ( messaggio_modulus contiene STATO_BASE );
END;
```

D.5: Livello MODULUS: struttura e descrizione dei comandi

Il livello MODULUS e´, finalmente, il livello applicativo del protocollo di comunicazione fra computer di controllo e BASE MODULUS. Livello applicativo significa che questo livello riguarda la specifica applicazione: cioe´ nel computer il programma di controllo di movimento del robot e nella BASE MODULUS i programmi del microprocessore che controlla i suoi motori .
Abbiamo gia´ visto e usato alcuni messaggi di questo livello nel paragrafo C.5; ora completiamo la nostra conoscenza .
I messaggi ammessi sono 10: 8 da computer verso la BASE e 2 dalla BASE del robot verso il computer di controllo. Vediamoli brevemente.

PC ----> BASE MODULUS

1. CAMBIO PARAMETRI: serve a cambiare i parametri di funzionamento standard dell´anello di regolazione di posizione e di velocita´ dei motori.

2. PARAMETRI POLINOMIO: comunica alla BASE i coefficienti del polinomio, che esprime la legge oraria, e altri parametri necessari all´esecuzione del movimento di uno o tutti e due i motori.

3. RICHIESTA STATO BASE: richiede alla BASE il suo stato di funzionamento.

4. RICHIESTA STATO BUFFER: richiede alla BASE lo stato dei "buffer di polinomio".

5. PENNA GIU´ (ACCESO): comanda l´abbassamento della punta della penna-plotter, per poter scrivere, o l´accensione dell´aspirapolvere.

6. PENNA SU (SPENTO): comanda il sollevamento da terra della punta della penna-plotter o lo spegnimento dell´aspirapolvere.

7. START MOTORE: comanda l´accensione di uno o di tutti e due i motori della BASE. Il motore eseguira´ il moto descritto dai parametri/polinomio ricevuti prima (e presenti nel buffer).

8. STOP MOTORE: comanda lo spegnimento di uno o di tutti e due i motori della BASE. Il buffer di parametri/polinomio relativo viene azzerato.

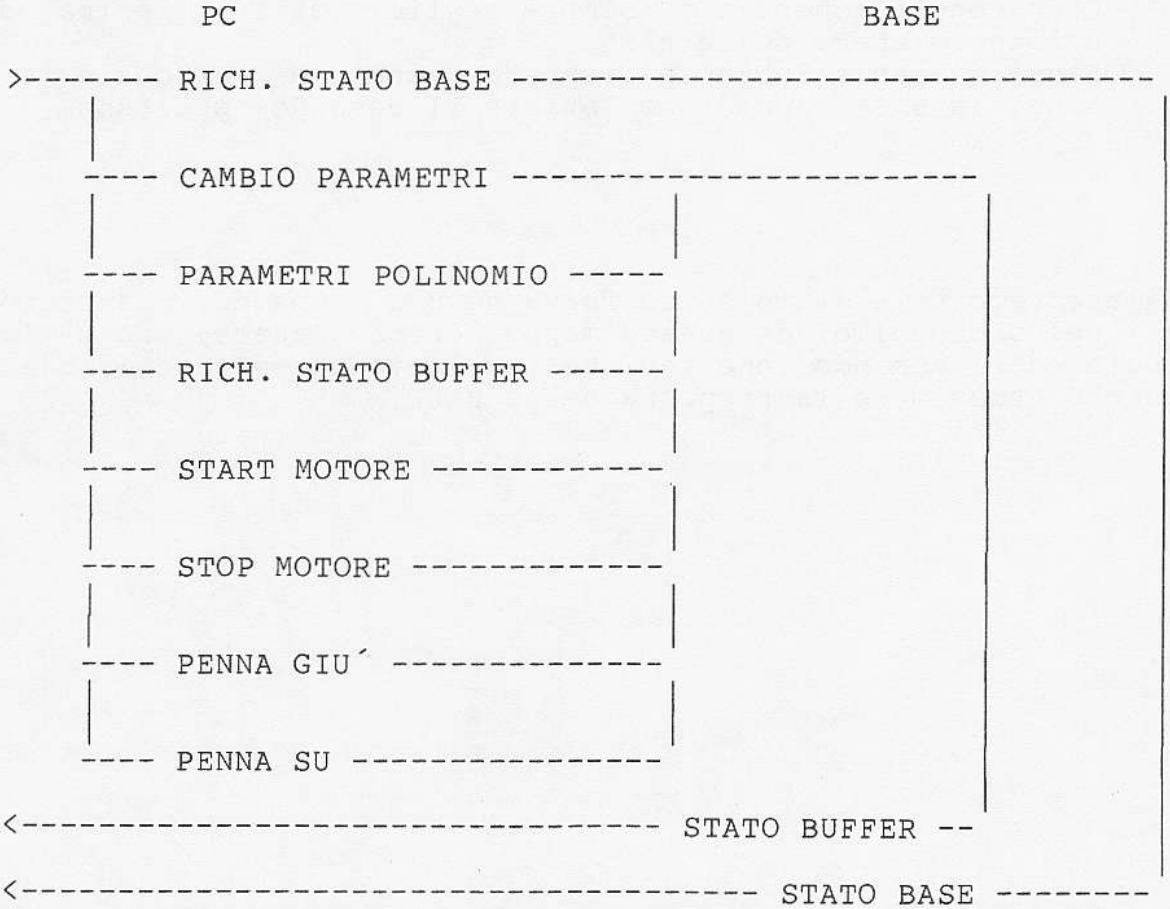
BASE MODULUS ----> PC

9. STATO BASE: e´ la risposta alla richiesta di stato fatta dal computer di controllo. Il messaggio comprende varie informazioni: errori di controllo, motori accesi/spenti, urti, stato buffer di polinomio, ...

10. STATO BUFFER: e´ la risposta alla richiesta di stato buffer (cioe´ buffer disponibile o occupato).
Ricordiamo che i questi buffer sono aree di memoria dove MODULUS ospita, finche´ non ha finito di eseguire, i parametri-polinomio di movimento ricevuti dal computer di controllo.

Oltre a questi due, puo´ arrivare anche un terzo tipo di messaggio al computer di controllo, che pero´ arriva direttamente dal microprocessore della RF-COMMAND KEYBOARD: lo vedremo nel par D.6 .

Fra qualche pagina vedremo in dettaglio il formato ed il contenuto di ogni messaggio.
Vediamo ora le regole per il colloquio a questo livello:



Questo diagramma non ci dice pero´ nulla sulla sequenze corrette con cui emettere i comandi. Vediamo le sequenze ammesse.

* La RICHIESTA DI STATO BASE e la RICHIESTA DI STATO BUFFER possono essere inoltrate in qualsiasi momento senza problemi.

* CAMBIO PARAMETRI invece deve essere inviato a BASE con motori fermi e, possibilmente, con i buffer vuoti. Se non si rispettano queste condizioni la BASE reagisce in modo imprevedibile.

  ATTENZIONE: SIRIUS non garantisce il funzionamento corretto del controllo di movimento se si cambiano i parametri standard.
  I parametri standard sono stati addottati dopo accurate prove in laboratorio e danno i risultati migliori.

* PENNA GIU´ / PENNA SU possono essere inviati in qualsiasi momento ma vengono eseguiti solo a BASE ferma; se la BASE si sta muovendo l´esecuzione avverra´ non appena il movimento e´ terminato.

* La RICHIESTA STATO BUFFER e´ utile quando si inviano polinomi raccordati, per sapere quando inviare il successivo polinomio (che deve arrivare prima che i motori si fermino perche´ hanno esaurito il moto previsto dal polinomio precedente).

* Come regola generale e´ sempre meglio richiedere prima di ogni azione lo stato della BASE.
  Fatto questo e letta la risposta, sappiamo in quale stato si trova la BASE e possiamo inviare il comando opportuno.

Le altre regole le diamo sotto forma di una mappa di transizione fra stati del protocollo: da questa mappa potete sapere, noto lo stato attuale ed il comando che inviate (o l´evento esterno), quale sara´ lo stato prossimo e la risposta della BASE.

COME LEGGERE LA MAPPA

Le frecce continue rappresentano transizioni dovute all´arrivo di comandi alla BASE.
Accanto ad ogni freccia e´ indicato il comando che ha causato la transizione ed il messaggio che la BASE invia in risposta (separato da /).
I messaggi che la BASE invia in risposta sono in realta´ composti da due parti: "ack" (o "nack") e "free" (o "busy").

"Ack" o "nack" appartengono al livello sottostante (quello logico) del protocollo di colloquio, che abbiamo gia´ visto nel paragrafo precedente.
Li abbiamo inclusi solo perche´, quando il buffer di polinomio e´ pieno, se si invia un altro polinomio si riceve un messaggio di buffer pieno con "nack".
In questo (e solo in questo) caso "nack" rappresenta un messaggio di questo livello di protocollo (applicativo: MODULUS).

"Free" (disponibile) o "busy" (occupato) sono i due stati del buffer di polinomio.

Nella mappa "POLI" sta per PARAMETRI POLINOMIO, "START" per START MOTORE, "STOP" per STOP MOTORE.

Non sono presenti i messaggi di PENNA SU e PENNA GIU´, CAMBIO PARAMETRI e RICHIESTA DI STATO BASE o BUFFER perche´ non influenzano gli stati rappresentati nella mappa: di fatto la mappa e´ solo un modello di funzionamento del sistema di controllo del motore e non comprende tutti i possibili "stati macchina" della BASE.
Quella reale risulterebbe complicatissima e inconsultabile.

Le frecce a puntini rappresentano transizioni avvenute per cause esterne diverse da un comando (urti, errori nei parametri, ...) o per cause interne (fine del movimento, ...).
Accanto ad ogni freccia a puntini e´ indicato l´evento che ha causato la transizione.

E a questo punto ... la mappa!

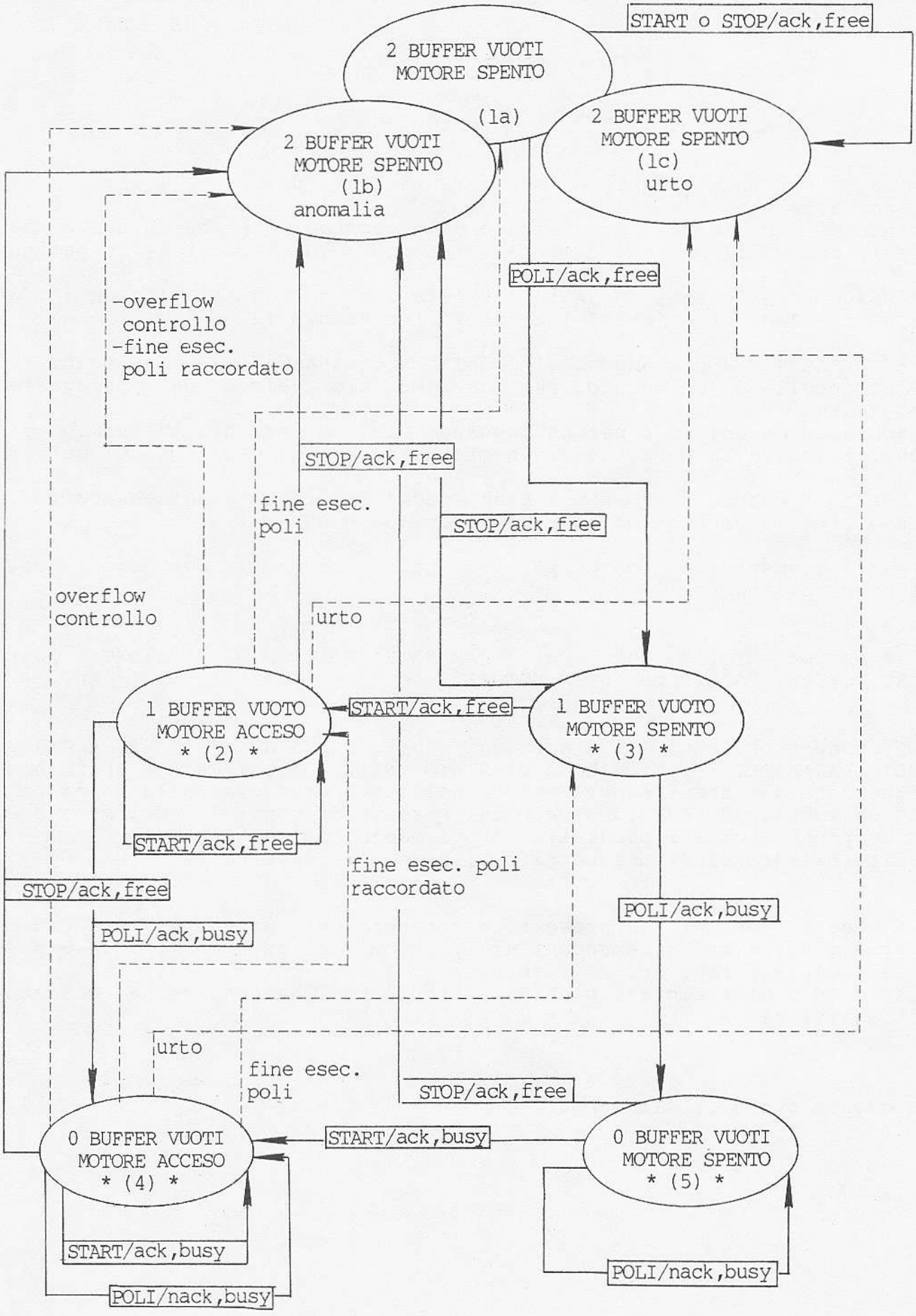

START o STOP/ack,free
2 BUFFER VUOTI
MOTORE SPENTO
(1a)
2 BUFFER VUOTI
MOTORE SPENTO
(1b)
anomalia
2 BUFFER VUOTI
MOTORE SPENTO
(1c)
urto
POLI/ack,free
-overflow
controllo
-fine esec.
poli raccordato
STOP/ack,free
fine esec.
poli
STOP/ack,free
overflow
controllo
urto
1 BUFFER VUOTO
MOTORE ACCESO
* (2) *
START/ack,free
1 BUFFER VUOTO
MOTORE SPENTO
* (3) *
START/ack,free
fine esec. poli
raccordato
STOP/ack,free
POLI/ack,busy
POLI/ack,busy
urto
fine esec.
poli
STOP/ack,free
0 BUFFER VUOTI
MOTORE ACCESO
* (4) *
START/ack,busy
0 BUFFER VUOTI
MOTORE SPENTO
* (5) *
START/ack,busy
POLI/nack,busy
POLI/nack,busy

fig d.5.1

Come avrete notato, noi abbiamo riportato solo la mappa per un motore. Quella dell´altro motore e´ del tutto identica.

Vedremo nella descrizione dei comandi che seguira´, che i comandi di START, STOP e POLINOMIO si possono inviare ad uno dei due motori o anche a tutti e due insieme.
E´ cosi´ possibile, per esempio, specificare un unico polinomio ai due motori senza doverlo inviare due volte, una per ogni motore.

Attenzione pero´ in questo caso se solo uno dei buffer di polinomio dei due motori e´ occupato riceveremo come risposta "nack,busy" ed il polinomio inviato verra´ scartato, anche se il buffer dell´altro motore fosse disponibile.

Inoltre, a proposito di "nack", se volessimo mischiare il livello inferiore di protocollo, quello logico, in questo schema, basterebbe aggiungere ad ogni stato delle frecce cosi´:

fig d.5.2

Cioe´ ad ogni comando/messaggio ricevuto male la BASE risponde con "nack" e non cambia stato.

Tornando a livello applicativo MODULUS, mancano alcune precisazioni, che non abbiamo incluso nella mappa per questioni di leggibilita´.

- Facciamo notare che se si invia un polinomio quando i buffer sono gia´ pieni (stati (4) e (5)), la BASE risponde in modo diverso, a seconda che il colloquio avvenga direttamente via cavo o attraverso la RF-CK.
  Se avviene via cavo, la BASE non risponde con alcun messaggio e permane nello stesso stato; se avviene via RF-CK, invece, il PC ricevera´ un "nack" in risposta.

- Lo stato di partenza e´ (1a).
  Gli stati (1b) e (1c) sono del tutto equivalenti a (1a), salvo che riportano traccia di un´avvenuta anomalia o urto.

- Qualsiasi transizione che porti allo stato (1b) o (1c) cancellera´ il contentuto dei buffer di polinomio.

- Una richiesta di stato BASE, fatta quando si e´ nello stato "urto" (1c), fa passare allo stato di partenza (1a).
  Cio´ implica che ogni altra richiesta di stato base non causera´ piu´ una risposta con un messaggio di stato contenente l´indicazione "urto".

- Invece, con una richiesta di stato BASE non si passa dallo stato di "anomalia" (1b) allo stato di partenza (1a).
  Cio´ significa che la segnalazione di "anomalia" rimane nel messaggio di stato e sparira´ solamente dopo l´invio di un nuovo polinomio e relativo start motore.

Riportiamo ora tutti i messaggi di questo livello di protocollo con una dettagliata spiegazione del loro formato e contenuto.

1. CAMBIO PARAMETRI

La struttura del messaggio di cambio parametri e´ la seguente:

MESSAGGIO := OP.CODE/GUADAGNO/INTEGRAZIONE/TA

- OP. CODE: il campo op. code (= codice operativo) identifica sia il comando di cambio parametri che il motore a cui esso si riferisce.
Si ottiene facendo la somma fra il codice operativo (12 dec o 0C hex) e il codice di motore (0 = motore sinistro, 1 = motore destro, 3 = motore destro e sinistro insieme).

- GUADAGNO: e´ il guadagno associato all´anello di regolazione di velocita´ dei motori. Il valore standard e´ 5, che assicura un guadagno unitario.
Si possono programmare valori da 0 a 15.
SIRIUS garantisce il buon funzionamento della BASE solo con il valore standard.

- INTEGRAZIONE: e´ il tempo di integrazione usato nell´anello di regolazione PID dei motori. Il valore standard e´ 3, e si possono programmare valori da 0 a 15.
SIRIUS garantisce il buon funzionamento della BASE solo con il valore standard.

- TA: e´ l´intervallo di tempo ogni cui l´anello di controllo della posizione dei motori verifica il conteggio dei "passi" percorsi con quelli previsti.
TA (Tempo di Aggiornamento) e´ espresso in multipli di 8,33 millisecondi.
Il valore standard e´ 5, che corrisponde a TA = 41,65 mS.
Si possono programmare valori da 0 a 255 (cioe´ da 0 a oltre 2 secondi).

2. PARAMETRI POLINOMIO

La struttura del messaggio di parametri/polinomio e´ la seguente:

MESSAGGIO := OP.CODE/A8/V/TF/TM/PFlow/PFhigh

- OP. CODE: il campo op. code (= codice operativo) identifica sia il comando di cambio parametri che il motore a cui esso si riferisce.
Si ottiene facendo la somma fra il codice operativo del comando (24 dec o 18 hex) e il codice di motore (0 = motore sinistro, 1 = motore destro, 3 = motore destro e sinistro insieme).

- A8: corrisponde al coefficiente di accelerazione (moltiplicato per 8) del polinomio che esprime la legge oraria di ogni ruota.
Per aver un buon controllo dei motori, il valore di A8 puo´ variare fra 4 e 28.

- V: corrisponde al coefficiente di velocita´ del polinomio che esprime la legge oraria di ogni ruota.
Per aver un buon controllo dei motori, il valore di V puo´ variare fra 4 e 28.

Sia A8 che V possono essere numeri positivi o negativi (positivo corrisponde ad avanti, negativo all´indietro).
I valori negativi si codificano ponendo a 1 il bit piu´ significativo del byte.
In pratica basta sommare al valore di V o A8 il numero 128 dec (o 80 hex).
Il polinomio che esprime la legge oraria e´:

$$S (T) = ((A8 * T^2) / 16) + (V * T)$$

- TF: il campo TF contiene il numero di unita´ di tempo (espressa in "grani" di TA, detti U.T.A.) prima della fine del movimento in cui il microprocessore controlla il raggiungimento della posizione finale. Inoltre il bit piu´ significatAivo del campo se e´ a 0 indica la fine del movimento se, invece, e´ a 1 indica l´esistenza di un raccordo con il successivo polinomio.

Inizio Controllo pos. finale = (TM - TF) unita´ di TA

Puo´ assumere valori compresi fra 0 e 127.

- TM: (Tempo di Movimento) e´ il tempo complessivo del movimento descritto dalla legge oraria vista sopra espresso in unita´ di TA (U.T.A.).
Puo´ assumere valori compresi fra 0 e 255.

- PFlow & PFhigh: il campo PF (Posizione Finale) contiene un valore di 16 bit che esprime il numero di unita´ di spostamento (U.È.P.) che il motore deve aver eseguito alla fine del movimento.
PF puo´ assumere valori positivi o negativi: i valori negativi vengono specificati ponendo il bit piu´ significativo ad 1, come per V ed A8.
Il campo PF, essendo il pacchetto livello MODULUS organizzato in byte, viene diviso in due sottocampi di un byte: PFlow e PFhigh, che contengono rispettivamente gli 8 bit piu´ significativi e gli 8 meno significativi dell´intero campo di 16 bit.
Va da se che il bit di segno rimarra´ automaticamente nel campo PFhigh.
In totale PF puo´ assumere valori compresi fra -32767 e +32767

3. RICHIESTA STATO BASE

La struttura del messaggio di richiesta stato BASE e´ la seguente:

MESSAGGIO := OP.CODE (0)

- OP. CODE: il campo op. code (= codice operativo) contiene il valore 0.

4. RICHIESTA STATO BUFFER

La struttura del messaggio di richiesta stato buffer e´ la seguente:

MESSAGGIO := OP.CODE (1)

- OP. CODE: il campo op. code (= codice operativo) contiene il valore 1.

5. PENNA GIU´ (ACCESO)

La struttura del messaggio di penna giu´ (o aspirapolvere acceso) e´ la seguente:

MESSAGGIO := OP.CODE (192 dec)

- OP. CODE: il campo op. code (= codice operativo) contiene il valore 192 dec o C0 hex .

6. PENNA SU´ (SPENTO)

La struttura del messaggio di penna su´ (o aspirapolvere spento) e´ la seguente:

MESSAGGIO := OP.CODE (193 dec)

- OP. CODE: il campo op. code (= codice operativo) contiene il valore 193 dec o C1 hex .

7. START MOTORE

La struttura del messaggio di start motore e´ la seguente:

MESSAGGIO := OP.CODE (80/81/83 dec)

- OP. CODE: il campo op. code (= codice operativo) comprende anche il motore a cui si riferisce il comando.
Si ottiene facendo la somma fra il codice operativo del comando (80 dec o 50 hex) e il codice di motore (0 = motore sinistro, 1 = motore destro, 3 = motore destro e sinistro insieme).

8. STOP MOTORE

La struttura del messaggio di stop motore e´ la seguente:

MESSAGGIO := OP.CODE (64/65/67 dec)

- OP. CODE: il campo op. code (= codice operativo) comprende anche il motore a cui si riferisce il comando.
Si ottiene facendo la somma fra il codice operativo del comando (64 dec o 40 hex) e il codice di motore (0 = motore sinistro, 1 = motore destro, 3 = motore destro e sinistro insieme).

9. STATO BASE

La struttura del messaggio di stato BASE e´ la seguente:

MESSAGGIO := ERR1low/ERR1high/ERR2low/ERR2high/ERR3low/ERR3high/ INIZIO-FINE/BYTE BUMPERS/STATO MOVIMENTO/STATO MOTORI

- ERR1low & ERR1high: il campo ERR1 (ERRore motore 1 - sinistro -) contiene un valore di 16 bit.
Durante il movimento del motore il campo contiene il numero di unita´ di spostamento (U.E.P.) compiuti dal motore 1 fino a quell´istante, rispetto al polinomio in esecuzione.
A movimento finito il campo contiene l´errore, in numero di unita´ di spostamento (U.E.P.), compiuto dal motore 1, a cui e´ stato aggiunto il valore 32768 dec o 8000 hex .
L´errore finale si calcola allora cosi´:

ERRORE FINALE 1 = ERR1 - 32768

La somma con 32768 si spiega considerando che questo campo viene inizializzato al valore di 32768 dec (o 8000 hex) ogniqualvolta inizia l´esecuzione di un polinomio non sia raccordato al precedente.
In totale ERR1 puo´ assumere valori compresi fra 0 e 65535, e dunque ERRORE FINALE 1 assume valori fra -32768 e +32767.

- ERR2low & ERR2high: il campo ERR2 (ERRore motore 2 - destro -) contiene un valore di 16 bit.
Il contenuto e´ il medesimo del campo ERR1, salvo che si riferisce al motore 2, cioe´ quello destro.

- ERR3low & ERR3high: il campo ERR3 (ERRore motore 3) contiene un valore di 16 bit.
Per la BASE, questo campo contiene sempre il valore 0.
E´ presente in quanto il sistema di controllo dei motori di MODULUS e´ in grado di gestire 3 motori contemporaneamente.
Ma nell´applicazione della BASE ci sono solo 2 motori, e dunque questo campo, e piu´ in generale tutti quelli che riguardano il terzo motore, saranno sempre lasciati a zero.

- INIZIO-FINE: e´ un campo di 8 bit che contiene le indicazioni di stato dei sensori di inizio corsa e di sovraccarico (fine corsa) dei motori.
Per la BASE, questo campo contiene sempre il valore 3F hex o 63 dec, in quanto i due motori della BASE non dispongono di controllo di inizio/fine corsa o sovraccarico.
E´ presente per generalita´, in quanto il sistema di controllo dei motori di MODULUS e´ in grado di gestire questi segnali per altre applicazioni.

- BYTE BUMPERS: e´ un campo di 8 bit che contiene le indicazioni di stato degli otto sensori di urto della BASE.
Normalmente ogni bit vale 0; quando un sensore ha rivelato l´urto, il corrispondente bit assume valore 1.
La corrispondenza con i sensori e´ la seguente (per corrispondenza grafica vedi par C.2).

bit 0	1 = urto SINISTRA (bumper 1)
bit 1	1 = urto AVANTI-SINISTRA (bumper 2)
bit 2	1 = urto AVANTI (bumper 3)
bit 3	1 = urto AVANTI-DESTRA (bumper 4)
bit 4	1 = urto INDIETRO-SINISTRA (bumper 5)
bit 5	1 = urto INDIETRO (bumper 6)
bit 6	1 = urto INDIETRO-DESTRA (bumper 7)
bit 7	1 = urto DESTRA (bumper 8)

- STATO MOVIMENTO: e´ un campo di 8 bit che contiene le indicazioni di stato del movimento dei motori della BASE.
Numeriamo i bit da 0 (bit meno significativo) a 7 (bit piu´ significativo) e vediamo il significato dei vari bit di stato.

bit 0	1 = motore 1 - sinistro - fermato a causa di una anomalia, di un´urto o di un comando di STOP. 0 = movimento normale.
bit 1	1 = motore 2 - destro - fermato a causa di una anomalia, di un´urto o di un comando di STOP. 0 = movimento normale.
bit 2	vale sempre 0; e´ l´analogo dei bit 0 o 1 per il terzo motore, che pero´ non e´ presente nella BASE.
bit 3	non significativo.
bit 4	se il motore 1 - sinistro - e´ stato fermato per anomalia, urto o stop (bit 0 = 1) allora 0 = valore contenuto nel campo ERR1 e´ riferito al penultimo polinomio inviato . 1 = valore contenuto nel campo ERR1 e´ riferito all´ultimo polinomio inviato alla BASE.
bit 5	se il motore 2 - destro - e´ stato fermato per anomalia, urto o stop (bit 0 = 1) allora 0 = valore contenuto nel campo ERR2 e´ riferito al penultimo polinomio inviato . 1 = valore contenuto nel campo ERR2 e´ riferito all´ultimo polinomio inviato alla BASE.
bit 6	vale sempre 0; e´ l´analogo dei bit 4 o 5 per il terzo motore, che pero´ non e´ presente nella BASE.
bit 7	non significativo.

- STATO MOTORI: e´ un campo di 8 bit che contiene le indicazioni di stato dei motori della BASE.
Numeriamo i bit da 0 (bit meno significativo) a 7 (bit piu´ significativo) e vediamo il significato dei vari bit di stato.

bit 0	1 = motore 1 - sinistro - in movimento. 0 = motore fermo.
bit 1	1 = motore 2 - destro - in movimento. 0 = motore fermo.
bit 2	vale sempre 0; e´ l´analogo per il motore 3, che pero´ non e´ presente nella BASE, dei bit 0 o 1.
bit 3	non significativo.
bit 4	0 = buffer motore 1 - sinistro - disponibile (free). 1 = buffer occupato (busy).
bit 5	buffer motore 2 - destro - disponibile (free). 1 = buffer occupato (busy).
bit 6	vale sempre 0; e´ l´analogo dei bit 4 o 5 per il terzo motore, che pero´ non e´ presente nella BASE.
bit 7	non significativo.

10. STATO BUFFER

La struttura del messaggio di stato buffer e´ la seguente:

MESSAGGIO := STATO BUFFER

- STATO BUFFER: e´ un campo di 8 bit che contiene le indicazioni di stato dei buffer dei motori della BASE, oltre che i messaggi di "conferma pacchetto ricevuto" del livello inferiore del protocollo (livello 3 = linea).
Numeriamo i bit da 0 (bit meno significativo) a 7 (bit piu´ significativo) e vediamo il significato dei vari bit di stato.

bit 0	1 = (NACK) messaggio livello linea: pacchetto precedente non ricevuto correttamente (in tal caso anche il bit 3 deve essere a 1).
bit 1	1 = (ACK) messaggio livello linea: pacchetto precedente ricevuto correttamente (in tal caso anche il bit 2 deve essere a 1).
bit 2	1 = (ACK) messaggio livello linea: pacchetto precedente ricevuto correttamente (in tal caso anche il bit 1 deve essere a 1).
bit 3	1 = (NACK) messaggio livello linea: pacchetto precedente non ricevuto correttamente (in tal caso anche il bit 0 deve essere a 1).
bit 4	0 = (FREE) buffer motore 1 - sinistro - disponibile. 1 = (BUSY) buffer occupato.
bit 5	0 = (FREE) buffer motore 2 - destro - disponibile. 1 = (BUSY) buffer occupato.
bit 6	vale sempre 0; e´ l´analogo dei bit 4 o 5 per il terzo motore, che pero´ non e´ presente nella BASE.
bit 7	non significativo.

D.6: Tastiera R.F. COMMAND KEYBOARD e controllo BASE MODULUS

Abbiamo ormai usato o visto tutte le funzioni svolte dalla RF-COMMAND KEYBOARD, o RF-CK.
Studiamone ora a fondo la struttura.

La RF-CK e' sostanzialmente composta da due blocchi: una TASTIERA INTELLIGENTE, ovvero una tastiera collegata ad un sistema a microprocessore, ed un MODEM RF che viene utilizzato sia dal computer di controllo che dalla tastiera "intelligente" per comunicare con MODULUS.

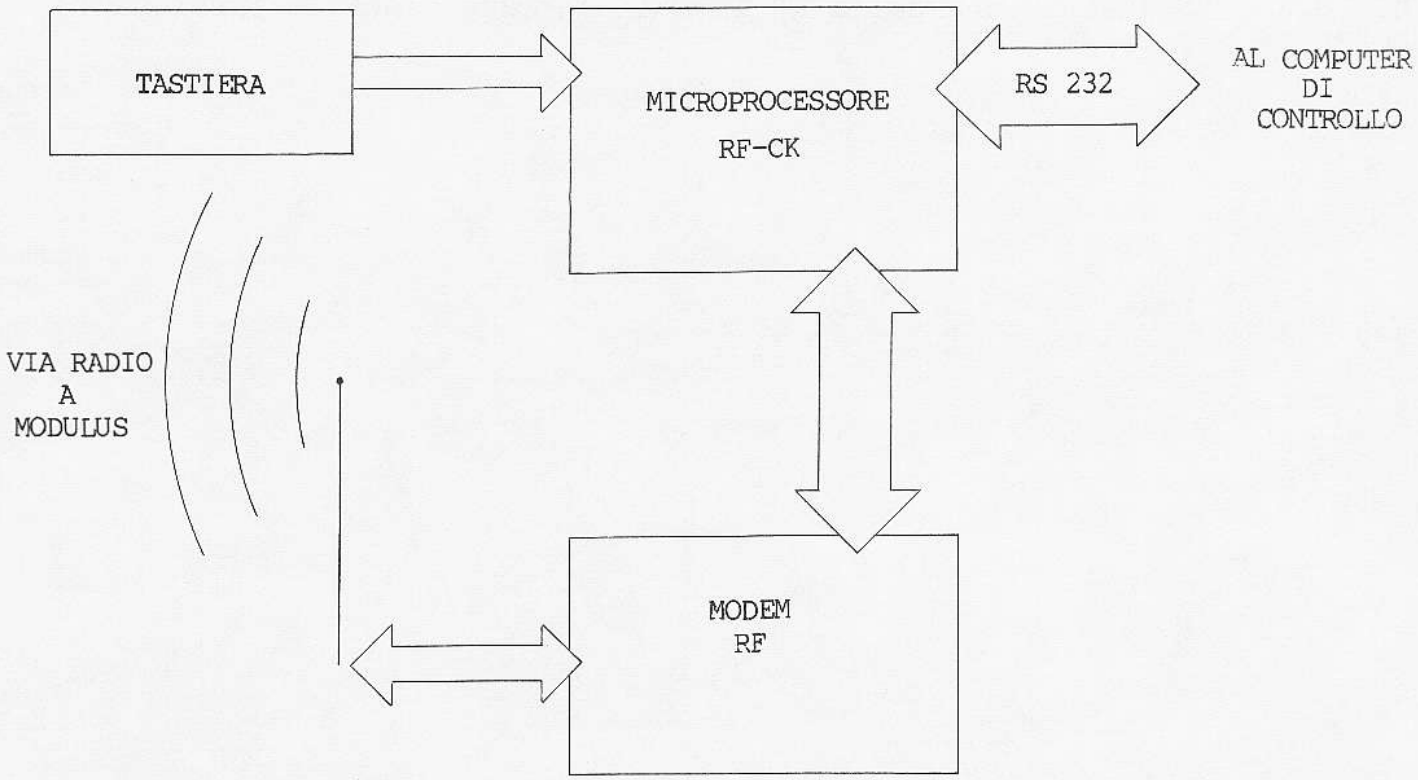


fig d.6.1

Questo e' chiaramente un modello semplificato: la divisione non e' in realta' cosi' netta.

La RF-CK e´ alimentata autonomamente da 8 batterie a stilo da 1,5 volt.
La tensione operativa e´ di 12 volt con assorbimento medio di 0,1 A.

E´ anche possibile alimentare dall´esterno la RF-CK utilzzando una apposita presa a cui si deve collegare un alimentatore stabilizzato che eroghi 0,2 A a 12 volt C.C. .

Per la descrizione dell´aspetto della RF-CK , delle operazioni di cambio batterie, alimentazione esterna, apertura, programmazione, ... fare riferimento al paragrafo A.4.2, A.4.3 .
Per la descrizione dei connettori, cavi e collegamenti consultare invece il paragrafo F.1 .

Dividiamo la descrizione della RF-CK nei suoi due aspetti funzionali: MODEM RF e TASTIERA INTELLIGENTE.

MODEM RF

Un modem e´ una apparecchiatura destinata a trasformare dei segnali digitali in una forma trasportabile a distanza: questo scopo viene raggiunto manipolando o modulando dei segnali analogici (tensioni, frequenze , correnti, campi elettromagnetici in genere - radio, luce, ... -).
Quando si riceve il segnale si deve compiere il lavoro inverso per "estrarre" l´informazione digitale dal segnale analogico o demodularlo (MODulatore-DEModulatore --> MODEM).
L´apparecchio che si incarica di rendere i dati trasportabili e gestirne il trasporto viene indicato con DCE (Data Communication Equipment).
Quello che fornisce e riceve i dati da trasportare viene indicato con DTE (Data Terminal Equipment) .

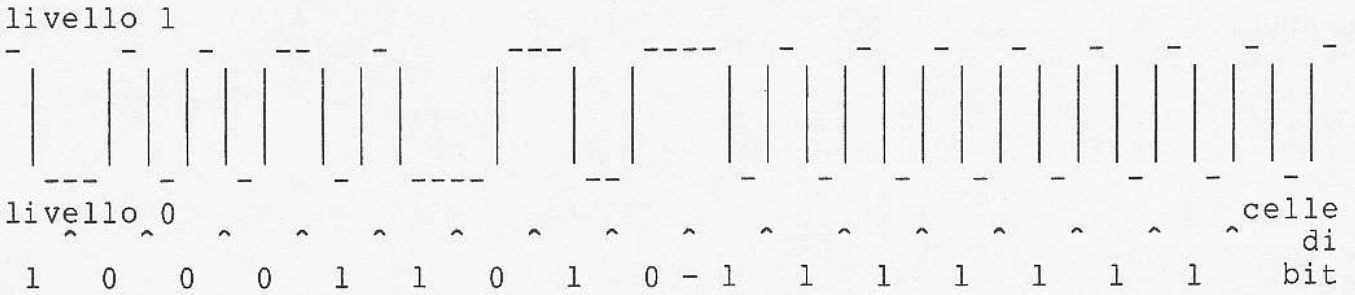
Chi ha letto il settore dedicato al "livello 1/RF" del paragrafo D.3 sa ormai quasi tutto sul MODEM RF.
Diamo qui una descrizione piu´ succinta.

Il modem opera su un canale half-duplex in radiofrequenza.
La distanza coperta e´ mediamente di 20 metri senza ostacoli fra RF-CK e MODULUS.
Pareti ed ostacoli, specie se metallici, riducono la portata utile e possono creare comunque delle "zone d´ombra".

Il modem opera in modulazione di ampiezza e usando, per i dati digitali, una codifica tipo Manchester.
La trasmissione e´ sincrona e avviene per pacchetti: i caratteri che la RF-CK o la BASE MODULUS trasmettono sono gia´ organizzati in messaggi o pacchetti.

Ogni pacchetto viene trasmesso sul canale RF preceduto da una sequenza di preavviso/sincronizzazione composta da 16 bit alternativamente a 0 e 1 che servono a dare il tempo al trasmettitore RF di generare un segnale stabile, seguita da una configurazione di sincronismo di inizio blocco formata da un bit e mezzo a livello 1 e un bit e mezzo a livello 0.
Segue il pacchetto in cui ad ogni byte viene aggiunto un nono bit che e´ normalmente a 0.
Il nono bit e´ a 1 solo nel primo byte del messaggio e nell´ultimo, che e´ sempre un byte di fine pacchetto contenente tutti 1 (11111111 bin o FF hex).
Viene inoltre trasmesso, nel mezzo byte (nibble) piu´ significativo del primo byte, il numero progressivo del pacchetto, con conteggio modulo 8.

Ogni bit del pacchetto viene trasmesso con codifica Manchester: un bit a 1 viene codificato con una transizione 1 --> 0, mentre un bit a 0 con una transizione 0 --> 1.
Vediamo come viene inviato il numero (lo chiameremo poi byte, essendo il numero contenuto in un byte appunto) 141 , che si suppone non il primo, seguito dal carattere di fine messaggio 255 (FF hex con nono bit a 1).
NB: 141 dec = 10110001 bin e 255 dec = 11111111 bin.



La "velocita´" di trasmissione su canale RF e´ di 300 baud con la BASE MODULUS e di 1200 baud con le configurazioni piu´ evolute di MODULUS (dotate di TECHNO-CAKE, ovvero di spicchioteca).
Questi baud rate garantiscono delle velocita´ massime di scambio dati fra MODEM e MODULUS rispettivamente di circa 30 e 120 byte/secondo.

L´interfaccia verso il computer di controllo e´ doppia: seriale RS 232 e seriale con livelli "Commodore".
La trasmissione qui e´ asincrona con caratteri di 8 bit dati, 1 bit di stop e senza controllo di parita´.
Si puo´ scegliere fra un baud rate di 300 o 9600 baud, che assicurano una velocita´ massima di scambio dati fra computer e modem di 30 e 960 byte/secondo, rispettivamente.

Il modem gestisce dei buffer, dei serbatoi di accumulo caratteri, per sincronizzare le differenti velocita´ di trasmissione del canale RF e quello RS 232 ed accumulare i caratteri asincroni dal computer per poterli spedire sincronicamente in pacchetti verso MODULUS.

Proprio per lo stesso motivo l´interfaccia verso il computer prevede un sistema di "handshake" sui fili RTS e CTS della RS 232.
I segnali presenti sul connettore dati della RF-CK sono TXD, RXD, RTS, CTS, GND implemetati secondo le specifiche RS 232 con l´RF-CK definito come DCE.

L´interfaccia a livelli "Commodore" prevede, sullo stesso connettore dati dell´RF-CK, tre segnali TX, RX, GND (senza handshake quindi su RTS e CTS) e solo alla velocita´ di 300 baud (scelta obbligata).
La velocita´ del canale RS 232 si programma su dip-switch ; vedi par A.4.3).

Il formato dei messaggi ed i protocolli fra RF-CK e computer di controllo sono descritti nei paragrafi D.4. e D.5 .

TASTIERA INTELLIGENTE

La RF-CK e´ dotata di una tastiera a membrana a 36 tasti collegata ad un microprocessore ad 8 bit.
Il microprocessore puo´ dialogare sia con la BASE MODULUS via modem RF sia con il computer di controllo attraverso l´interfaccia seriale.

La tastiera serve a comandare direttamente la BASE MODULUS (modo operativo 1; vedi paragrafo A.4.2) emettendo sequenze di comando a livello "MODULUS" oppure come organo di ingresso, una tastiera supplementare, del computer di controllo (modo operativo 2) o del computer/CPU 16 bit a bordo di MODULUS (modo operativo 3).
Il modo operativo della RF-CK si seleziona ogni volta all´accensione.

Se, mentre si accende, si tiene premuto un tasto qualsiasi allora si seleziona il modo 1. La RF-CK conferma la selezione del modo 1 emettendo tre "bip" consecutivi.

Se, invece, si accende senza tenere premuto alcun tasto la RF-CK seleziona il modo operativo 2 o 3 a seconda della predisposizione interna (selezionabile all´interno della RF-CK muovendo alcuni interruttori di programmazione; vedi paragrafo A.4.3).

Trattiamo separatamente i 3 modi operativi.

MODO 1: comando diretto della BASE

Nel modo 1 undici tasti della tastiera sono dedicati al controllo diretto delle operazioni della BASE MODULUS.
Tali funzioni sono state esplorate nel corso del capitolo B.

Siamo ora in grado, con le nozioni acquisite nei capitoli C e D, di comprenderle a fondo.

1. TASTI AVANTI/INDIETRO VELOCE/LENTO:

La sequenza operativa adottata dalla RF-CK e' la seguente:

```
BEGIN
   REPEAT
        ascolta_tastiera;
   UNTIL ( tasto_movimento premuto );

   trasmetti_un_pacchetto ( contenente "polinomio raccordato" );
   trasmetti_un_pacchetto ( contenente "polinomio raccordato" );
   trasmetti_un_pacchetto ( contenente "start motore" );
   ricevi_un_pacchetto;
   estrai_messaggio_modulus;

   WHILE ( tasto_movimento premuto )
   DO BEGIN
          IF ( messaggio_modulus = buffer-free )
          THEN BEGIN
                    trasmetti_un_pacchetto ( contenente "polinomio
                                             raccordato" )
                    trasmetti_un_pacchetto ( contenente "start
                                             motore" )
                    ricevi_un_pacchetto;
                    estrai_messaggio_modulus;
          END;

          ELSE BEGIN
                    ascolta_tastiera;
          END;
   END;

   trasmetti_un_pacchetto ( contenente "stop motore" );
END;
```

La sequenza e' una versione semplificata della vera procedura (in realta' la CPU della RF-CK deve gestire svariati eventi e compiti in contemporanea con tecniche di multi-tasking).

Il polinomio inviato riguarda il motore destro o sinistro o tutti e due, a seconda dei tasti premuti contemporaneamente sulla tastiera. Lo stesso vale per i comandi di start/stop motore.

Riportiamo qui i parametri/polinomi usati dalla CPU della RF-CK:

```
AVANTI VELOCE   (sinistro): 24,0,20,128,255,236,19
AVANTI LENTO    (sinistro): 24,0,10,128,255,246,9
INDIETRO LENTO  (sinistro): 24,0,138,128,255,246,137
INDIETRO VELOCE (sinistro): 24,0,148,128,255,236,147

AVANTI VELOCE   (destro): 25,0,20,128,255,236,19
AVANTI LENTO    (destro): 25,0,10,128,255,246,9
INDIETRO LENTO  (destro): 25,0,138,128,255,246,137
INDIETRO VELOCE (destro): 25,0,148,128,255,236,147
```

Se avete provato a decifrarli, facilmente avrete trovato delle difficolta´.
Molte delle difficolta´ nella lettura di questi polinomi/parametri vengono dal fatto che le unita´ di informazione che li compongono non sono sempre i byte.
Diventa molto piu´ naturale leggerli se li riscriviamo bit a bit, in binario, o esadecimale.
A titolo di esempio decodifichiamo insieme questi parametri/polinomio:

INDIETRO LENTO (sinistro): 24,0,138,128,255,246,137

esadecimale : 18,00,8A,80,FF,F6,89

binario :
00011000,00000000,10001010,10000000,11111111,11110110,10001001

Usiamo le unita´ di misura definite nel capitolo C (UEP e UTA).

- CODICE OPERATIVO: 24 + 0 (24=sinistro, 25=destro, 27=sinistro e destro) (= parametri/polinomio sono per motore sinistro)

- ACCELERAZIONE: 0

- VELOCITA´: 8A hex = 10001010 = - A hex (velocita´ -10 UEP/UTA)
  NB il bit di peso maggiore da il segno: se e´ 1 allora il numero e´ negativo.

- TEMPO FINE & RACCORDO: 80hex = 10000000 (si raccordo; non c´e´ "tempo fine" perche´ il movimento non termina ma si raccorda col successivo)
  NB il bit di peso maggiore dice se il polinomio e´ raccordato (si = 1)

- TEMPO MOVIMENTO: 255 dec (tempo totale di movimento con il polinomio specificato = 255 UTA)

- PUNTO FINALE (parte bassa): F6 hex (= 246 UEP)
- PUNTO FINALE (parte alta): 89 hex = 10001010 = -9 hex (= -9 * 256 UEP)

  TOTALE PUNTO FINALE: (-9 * 256) + 246= 2550 UEP

Notate che qui VELOCITA´ * TEMPO MOVIMENTO= PUNTO FINALE.
Questo significa che la BASE non dovra´ fare uso del controllo finale fine di posizione, che permette di raggiungere posizioni finali che non siano multipli interi della velocita´ (ricordate che tutti i campi sono numeri interi!).

NOTA: questi messaggi sono quelli del protocollo livello "MODULUS": la CPU della RF-CK aggiungera´ tutti i campi necessari per comporre il pacchetto da inviare a MODULUS (vedi paragrafo D.3).
Se volete usare questi polinomi/parametri dovete aggiungere anche voi i campi previsti dal protocollo livello linea.
Usiamo il solito polinomio di esempio; questo verra´ inviato dal computer sulla linea RS 232 cosi´:

hex: 08, 00, 18, 00, 8A, 80, FF, F6, 89, 9A

dec: 8, 0, 24, 0,138,128,255,246,137,154

numero byte (escluso LRC) | identif. slave | LRC

2. TASTI PU e PD

Questi tasti causano semplicemente l´emissione di una comando di PENNA SU (Pen Up) e PENNA GIU´ (Pen Down); vedi paragrafo D.5 .
I messaggi a livello "MODULUS" sono:

PENNA SU : 193 dec (= C1 hex)

PENNA GIU´: 192 dec (= C0 hex)

NOTA: per lo stesso discorso di prima, il pacchetto che dovrete inviare su linea seriale per trasmettere PENNA SU sara´:

hex: 02, 00, C1, C3

dec: 2, 0,193,195

2. TASTO SR

Questo tasto causa l´emissione di una RICHIESTA DI STATO BASE; vedi paragrafo D.4 e D.5 .
Il messaggio a livello "MODULUS" e´:

RICH. STATO : 0 dec (= 0 hex)

NOTA: per lo stesso discorso di prima, il pacchetto che dovrete inviare su linea seriale per trasmettere RICHIESTA DI STATO BASE sara´:

hex: 02, 00, 00, 02

dec: 2, 0, 0, 2

MODO 2: comando MODULUS da computer esterno

Nel modo 2 la tastiera e´ al servizio del computer esterno di controllo connesso alla RF-CK via interfaccia seriale.
La RF-CK invia al computer, via i/f seriale appunto, l´indicazione di tasto premuto.

Sara´ il programma che "gira" sul computer stesso ad assegnare un significato ad ogni tasto della RF-CK.
Per questo SIRIUS fornisce delle mascherine applicabili alla parte destra e sinistra della tastiera: potrete scrivere su tali mascherine il nuovo significato assegnato ad ogni tasto.
La parte centrale della tastiera e´ gia´ marcata, per comodita´, come una tastiera numerica.

La RF-CK segnala l´avvenuta pressione di un tasto inviando al computer un messaggio di "TASTO" di un byte.
Questo byte contiene un valore compreso fra 48 e 83 dec (30 - 53 hex) ed e´ quindi distinguibile da un messaggio di risposta proveniente direttamente da MODULUS, che nel primo byte (ID. SLAVE) contiene un valore compreso fra 0 e 15 dec (0 - F hex).

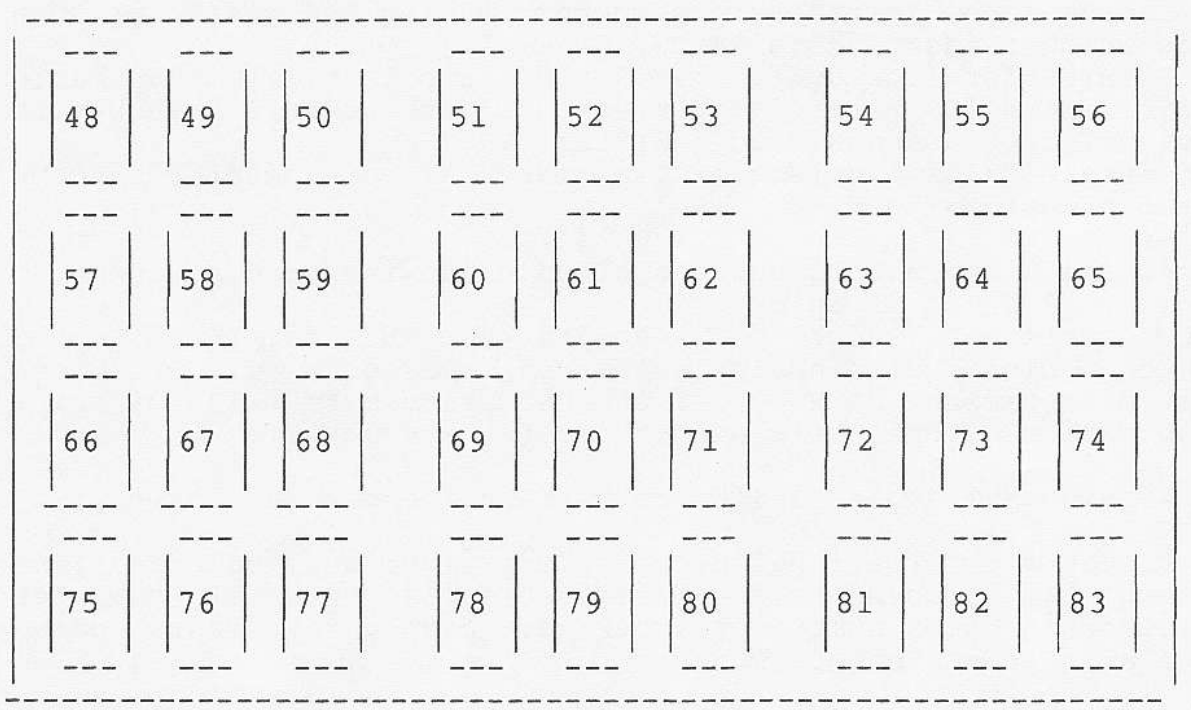
Questo messaggio di TASTO e´ del tutto particolare per due motivi.

Primo non rispetta i formati del protocollo a livello linea descritti nel paragrafo D.4, in quanto viene trasmesso dalla RF-CK al computer di controllo senza campo numero caratteri di messaggio, senza campo messaggio e anche senza campo LRC.

Secondo, mentre i messaggi da MODULUS arrivano sempre in risposta a messaggi del computer di controllo, il messaggio di TASTO arriva nel momento in cui viene premuto un tasto della RF-CK.

ATTENZIONE: questo implica che il programma che "gira" sul vostro computer deve ascoltare la linea seriale RS 232 anche quando non ha inviato alcun comando verso la RF-CK/MODULUS.
Inoltre, nell´esame dei messaggi ricevuti, deve saper sempre distinguere il messaggio di TASTO dagli altri .
Questo in qualsiasi situazione del protocollo; per esempio: il vostro programma invia un pacchetto contenete un polinomio alla BASE e si attende un pacchetto contenente conferma.
Ma nessuno vieta che qualcuno prema un tasto della RF-CK e la RF-CK invii tempestivamente il messaggio di TASTO, prima del pacchetto proveniente dalla BASE.
Il programma che realizza il protocollo deve saper gestire questo tipo di situazioni senza perdere ne´ scartare alcuno dei due messaggi: quello di TASTO che non si aspettava e quello di conferma che si aspettava subito, ma e´ arrivato dopo.

I valori-codice contenuti nel messaggio-byte di TASTO sono assegnati ai tasti della tastiera in ordine crescente dall´alto verso il basso e da sinistra verso destra (cioe´ nello stesso ordine in cui stiamo leggendo queste parole), partendo dal valore 48 dec.
Ma se avete ancora dei dubbi siete "autorizzati" a dare un occhiata allo schemino sottostante.



MODO 3: comando da CPU 16 bit a bordo di MODULUS

Questo modo operativo non e´ utilizzabile con la sola BASE MODULUS, in quanto la CPU a 16 bit puo´ essere installata solo su MODULUS dotato di TECHNO-CAKE (o spicchioteca: vedi paragrafo A.1).
In ogni caso vale esattamente il discorso fatto per il modo 2, ma questa volta i messaggi di TASTO vengono inviati, in un opportuno formato e attraverso il canale RF, alla CPU a bordo di MODULUS.

D.7: Come realizzare il protocollo

Diamo ora una descrizione delle procedure per realizzare il colloquio con MODULUS fino a livello 3, cioe´ linea/rete.

Un esempio di gestione del protocollo livello 4, applicativo MODULUS, e´ stato descritto nel paragrafo C.5.3.
Quella gestione del livello 4 fa uso delle procedure a livello 3 che descriveremo in questo paragrafo.

Nel paragrafo E.1 verranno date istuzioni precise per tradurre le procedure del paragrafo C.5.3 e di questo paragrafo in programma BASIC.

Usiamo una descrizione delle procedure che utilizza notazioni e funzioni pseudo-BASIC (ASC,CHR$, LEFT$, ...) e strutture di controllo pseudo-PASCAL.
Il risultato e´ un linguaggio strano, ma intuitivo e chiaro.

La traduzione delle strutture di controllo pseudo-PASCAL in BASIC viene descritta nel paragrafo E.1 .

Useremo una descrizione top-down a due livelli: il primo piu´ descrittivo ed il secondo piu´ operativo e dettagliato.

Riportiamo prima le procedure in una vesione piu´ semplice, che non prevede l´utilizzo della tastiera della RF-CK, e poi la versione piu´ complicata, che prevede la gestione dei messaggi di "tasto" dalla RF-CK.

AVVERTIAMO CHE LE PROCEDURE DESCRITTE SONO CORRETTE, MA SIRIUS GARANTISCE ED ASSISTE SOLO IL SOFTWARE ACQUISTATO SU SUPPORTI MAGNETICI ED ELETTRONICI CON IL PROPRIO MARCHIO.

1. PROTOCOLLO SENZA GESTIONE TASTIERA RF-CK

Abbiamo gia´ individuato, in fondo al paragrafo D.4, le due procedure che ci servono:

- inoltra_comando : questa si occupa di inoltrare un qualsiasi messaggio/comando a MODULUS, salvo la "richiesta di stato BASE".

- inoltra_richiesta_stato : questa invece si occupa di inoltrare la richiesta di stato BASE MODULUS e poi riceverlo.

```
inoltra_comando (IN:comando$;OUT:stato_buffer);

BEGIN
    REPEAT
         trasmetti_pacchetto(IN:comando$;OUT:-);

         ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

         estrai_livello_modulus(IN:pacchetto_risposta$;OUT:risposta$);

         separa_livello_linea(IN:risposta$;OUT:risposta_linea);

         separa_livello_modulus(IN:risposta$;OUT:stato_buffer);
    UNTIL (risposta_linea = ack);

    OUTPUT stato_buffer;
END;
```

Vediamo ora il livello dettagliato:

```
inoltra_comando (IN:comando$;OUT:stato_buffer);

BEGIN
    ack% := 6;
    nack% := 9;

    REPEAT
        trasmetti_pacchetto(IN:comando$;OUT:-);

        ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

        risposta$ := MID$(pacchetto_risposta$,2,
                     LEN(pacchetto_risposta)-2);

        IF ( LEN(risposta$) = 1 )
        THEN BEGIN
                stato_buffer% := ASC(risposta$) AND 240;

                risposta_linea% := ASC(risposta$) AND 15;
        END;

        ELSE BEGIN
                risposta_linea% := nack%;
        END;
    UNTIL ( risposta_linea% = ack );

    OUTPUT stato_buffer%;
END;
```

Il dettaglio delle procedure trasmetti_pacchetto e ricevi_pacchetto lo vedremo fra poco.
Vediamo prima la inoltra_richiesta_stato .

```
inoltra_richiesta_stato (IN:-;OUT:stato_base$);

BEGIN
    REPEAT
        trasmetti_pacchetto(IN:comando_richiesta_stato$;OUT:-);

        ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

        estrai_livello_modulus(IN:pacchetto_risposta$;OUT:risposta$);

    UNTIL ( LEN(risposta$) = 10 );

    stato_base$ := risposta$;

    OUTPUT stato_base$;
END;
```

Vediamo ora il livello dettagliato:

```
inoltra_richiesta_stato (IN:-;OUT:stato_base$);

BEGIN
    comando_richiesta_stato$ := CHR$(0);

    REPEAT
         trasmetti_pacchetto(IN:comando_richiesta_stato$;OUT:-);

         ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

         risposta$ := MID$(pacchetto_risposta$,2,
                      LEN(pacchetto_risposta)-2);

    UNTIL ( LEN(risposta$) = 10 );

    stato_base$ := risposta$;

    OUTPUT stato_base$;
END;
```

Ecco il dettaglio delle procedure trasmetti/ricevi_pacchetto.

```
trasmetti_pacchetto (IN:messaggio$;OUT:-);

BEGIN
    cavo% := 0; radio% := 1;

    mezzo% := radio%; (SE USATE CONNESSIONE VIA CAVO ALLORA
                       mezzo := cavo)

    fine_messaggio$ := CHR$(255);

    id_slave$ := CHR$(0);

    pacchetto$ := id_slave$ + messaggio$;

    IF ( mezzo% = radio% )
    THEN BEGIN
            pacchetto$ := CHR$( LEN(pacchetto$) ) + pacchetto$;

            calcola_lrc(IN:pacchetto$;OUT:lrc$);

            pacchetto$ := pacchetto$ + lrc$;

    END;

    ELSE BEGIN
            calcola_lrc(IN:pacchetto$;OUT:lrc$);

            pacchetto$ := pacchetto$ + lrc$;

            pacchetto$ := pacchetto$ + fine_messaggio$;
    END;

    trasmetti_RS232(IN:pacchetto$;OUT:-);
END;
```

La procedura trasmetti_RS232 spedisce sull´interfaccia RS 232 la stringa di caratteri "pacchetto$".
Di solito corrisponde ad una funzione/istruzione di sistema del linguaggio o del calcolatore che usate, e quindi non la dettagliamo ulteriormente.

In questo caso la procedura di ricezione accetta solo pacchetti che inizino con l´identificatore della BASE MODULUS: i messaggi di "tasto" vengono ignorati.

```
ricevi_pacchetto (IN:-;OUT:pacchetto$);

BEGIN
    id_slave$:= CHR$(0);

    REPEAT

        REPEAT
            ricevi_RS232(IN:-;OUT:carattere$);
        UNTIL ( carattere$ = id_slave$ );

        pacchetto$ := carattere$;

        ricevi_RS232(IN:-;OUT:carattere$);

        pacchetto$ := pacchetto$ + carattere$;

        lunghezza_messaggio% := ASC(carattere$);

        FOR n%=1 TO lunghezza_messaggio%
        DO BEGIN
            ricevi_RS232(IN:-;OUT:carattere$);

            pacchetto$ := pacchetto$ + carattere$;
        END;

        ricevi_RS232(IN:-;OUT:carattere$);

        lrc_ricevuto$:= carattere$;

        calcola_lrc(IN:pacchetto$;OUT:lrc$);
    UNTIL ( lrc_ricevuto$ = lrc$ );

    OUTPUT pacchetto$;
END;
```

La procedura ricevi_RS232 riceve sull´interfaccia RS 232 un carattere "carattere$".
Di solito corrisponde ad una funzione/istruzione di sistema del linguaggio o del calcolatore che usate, e quindi non la dettagliamo ulteriormente.

E´ buona regola legare la ricezione di un carattere da linea RS 232 ad un tempo massimo di attesa, scaduto il quale il programma riprende l´esecuzione considerando errata la risposta ricevuta (cioe´ risposta non ricevuta entro tempo max = risposta errata).

Siccome l'implementazione di tale meccanismo (time-out) e' strettamnte dipendente dal computer e dal linguaggio che usate, noi non l'abbiamo usato nella procedura appena descritta.

Rimane da definire la procedura calcola_lrc .

```
calcola_lrc (IN:messaggio$;OUT:lrc$);

BEGIN
    lrc% := 0;

    FOR k%=1 TO LEN(messaggio$)
    DO BEGIN
           kappesimo_carattere$ := MID$(messaggio$,k%,1);

           kappesimo_valore% := ASC(kappesimo_carattere$);

           lrc% := lrc% XOR kappesimo_valore;
    END;

    lrc$ := CHR$(lrc%);

    OUTPUT lrc$;
END;
```

ATTENZIONE:

Segnaliamo che la procedura inoltra_comando va usata con cautela quando contemporaneamente si usa la connessione via CAVO e si inviano POLINOMI.
Infatti, quando il buffer di polinomio e' pieno, nella versione via cavo la BASE non risponde con alcun messaggio ad un successivo polinomio: la inoltra_comando rimarrebbe cosi' bloccata in attesa di un messaggio, bloccando tutto il resto del programma.
Le soluzioni sono due: o si chiede sempre lo stato buffer prima di inviare un polinomio, in modo da inviarlo solo quando il buffer non e' pieno, o si implementa il time-out di ricezione descritto sopra per la procedura di ricevi_pacchetto.

2. PROTOCOLLO CON GESTIONE TASTIERA RF-CK

Diamo ora la versione delle procedure inoltra-comando e inoltra-richiesta-stato che presta attenzione a eventuali pressioni dei tasti della tastiera della RF-CK.
Vengono ritornati, a chi chiama le procedure, i codici di tasto, dopo pero´ che ´ sono state eseguite le funzioni delle procedure stesse (cioe´ inoltro dei messaggi a livello MODULUS al robot).

Diamo poi delle procedure che, rispettivamente, attendono o controllano la pressione di un tasto della tastiera RF-CK.
Sono di fatto le equivalenti delle funzioni BASIC INPUT# e GET# sul "file" tastiera RF-CK.

Segnaliamo che in questo manuale non sono riportate procedure di protocollo a livello 4 che utilizzino la tastiera della RF-CK.

Potete facilmente inventarne voi, per esempio creando un livello 4 che fermi il movimento del robot alla pressione di un tasto qualsiasi della RF-CK oppure rifacendo le funzioni di comando diretto della BASE integrate nella RF-CK, quando opera in modo 1, operando pero´ in modo 2 e aggiungendo altre funzioni utili alla tastiera RF-CK.

```
inoltra_comando (IN:comando$;OUT:stato_buffer,tasti_premuti$);

BEGIN
   REPEAT
      trasmetti_pacchetto(IN:comando$;OUT:-);

      ricevi_pacchetto_&_tasto(IN:-;OUT:pacchetto_risposta$,
                               tasti_premuti$);

      estrai_livello_modulus(IN:pacchetto_risposta$;OUT:risposta$);

      separa_livello_linea(IN:risposta$;OUT:risposta_linea);

      separa_livello_modulus(IN:risposta$;OUT:stato_buffer);
   UNTIL (risposta_linea = ack);

   OUTPUT stato_buffer,tasti_premuti$;
END;
```

Vediamo ora il livello dettagliato:

```
inoltra_comando (IN:comando$;OUT:stato_buffer,tasti_premuti$);

BEGIN
    ack% := 6;
    nack% := 9;
    stringa_vuota$ := "";
    tasti_premuti$ := stringa_vuota$;
    id_base$ := CHR$(0);

    REPEAT
        trasmetti_pacchetto(IN:comando$;OUT:-);

        REPEAT
            ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

            id_slave$ := LEFT$(pacchetto_ricevuto$,1);

            IF ( id_slave$ <> id_base$)
            THEN BEGIN
                    tasti_premuti$ := tasti_premuti$ + id_slave;
            END;
        UNTIL ( id_slave$ = id_base$ );

        risposta$ := MID$(pacchetto_risposta$,2,
                     LEN(pacchetto_risposta)-2);

        IF ( LEN(risposta$) = 1 )
        THEN BEGIN
                stato_buffer% := ASC(risposta$) AND 240;

                risposta_linea% := ASC(risposta$) AND 15;
        END;

        ELSE BEGIN
                risposta_linea% := nack%;
        END;
     UNTIL ( risposta_linea% = ack );

     OUTPUT stato_buffer%,tasti_premuti$;
END;
```

Il dettaglio delle procedure trasmetti_pacchetto e ricevi_pacchetto lo vedremo fra poco.

Vediamo prima la inoltra_richiesta_stato .

```
inoltra_richiesta_stato (IN:-;OUT:stato_base$);

BEGIN
    REPEAT
         trasmetti_pacchetto(IN:comando_richiesta_stato$;OUT:-);

         ricevi_pacchetto_&_tasto(IN:-;OUT:pacchetto_risposta$,
                                  tasti_premuti$);

         estrai_livello_modulus(IN:pacchetto_risposta$;OUT:risposta$);
    UNTIL ( LEN(risposta$) = 10 );

    stato_base$ := risposta$;

    OUTPUT stato_base$,tasti_premuti$;
END;
```

Vediamo ora il livello dettagliato:

```
inoltra_richiesta_stato (IN:-;OUT:stato_base$);

BEGIN
    comando_richiesta_stato$ := CHR$(0);
    id_base$ := CHR$(0);

    REPEAT
         trasmetti_pacchetto(IN:comando_richiesta_stato$;OUT:-);

         REPEAT
              ricevi_pacchetto(IN:-;OUT:pacchetto_risposta$);

              id_slave$ := LEFT$(pacchetto_ricevuto$,1);

              IF ( id_slave$ <> id_base$)
              THEN BEGIN
                        tasti_premuti$ := tasti_premuti$ + id_slave;
              END;
         UNTIL ( id_slave% = id_base% );

         risposta$ := MID$(pacchetto_risposta$,2,
                      LEN(pacchetto_risposta)_2);

    UNTIL ( LEN(risposta$) = 10 );

    stato_base$ := risposta$;

    OUTPUT stato_base$,tasti_premuti$;
END;
```

Ecco il dettaglio della procedura ricevi_pacchetto:

```
ricevi_pacchetto (IN:-;OUT:pacchetto$);

BEGIN
    vero% := 1;
    falso% := 0;
    id_base$:= CHR$(0);
    ricevuto% := falso%;

    REPEAT
        ricevi_RS232(IN:-;OUT:carattere$);

        IF ( carattere$ = id_base$ )
        THEN BEGIN
                REPEAT
                    pacchetto$ := carattere$;

                    ricevi_RS232(IN:-;OUT:carattere$);

                    pacchetto$ := pacchetto$ + carattere$;

                    lunghezza_messaggio% := ASC(carattere$);

                    FOR n%=1 TO lunghezza_messaggio%
                    DO BEGIN
                        ricevi_RS232(IN:-;OUT:carattere$);

                        pacchetto$ := pacchetto$ + carattere$;
                    END;

                    ricevi_RS232(IN:-;OUT:carattere$);

                    lrc_ricevuto$:= carattere$;

                    calcola_lrc(IN:pacchetto$;OUT:lrc$);
                UNTIL ( lrc_ricevuto$ = lrc$ );

                ricevuto% := vero%;
        END;

        ELSE BEGIN

                id_slave% := ASC(carattere$);

                IF ( (id_slave% >=48) AND (id_slave% <= 83) )
                THEN BEGIN
                        pacchetto$ := carattere$;

                        ricevuto% := vero;
                END;
        END;
    UNTIL ( ricevuto% = vero% );
END;
```

Notate che in questo caso la procedura di ricezione accetta sia pacchetti che inizino con l´identificatore della BASE MODULUS che pacchetti/messaggi di "tasto".

La procedura trasmetti_pacchetto e´ la stessa gia´ descritta nella versione senza gestione tastiera.

Il dettaglio della procedura calcola_lrc e´ quello gia´ visto nella versione senza gestione della tastiera della RF-CK.

Valgono anche le stesse osservazioni fatte sulla ricevi_pacchetto senza gestione della tastiera (vedi pag 7 di questo paragrafo).

E SOFTWARE

E.1: Controllo da BASIC standard

In questo paragrafo trattiamo il modo per tradurre in programmi BASIC le procedure/flussi che abbiamo usato nei paragrafi C.5.2, C.5.3 e D.7 .
I programmi che ne ricaverete vi metteranno in grado di comandare, attraverso il vostro computer, sequenze di translazioni e rotazioni alla BASE MODULUS.

Le differenze fra lo pseudo-linguaggio che abbiamo usato per descrivere le procedure ed il BASIC sono sostenzialmente queste:

1. Il BASIC non possiede le stesse strutture di controllo che abbiamo usato noi.
   Non c´e´ per esempio REPEAT ... UNTIL (condizione), WHILE (condizione) DO ..., ecc.

2. Il BASIC non ammette nomi per sottoprogrammi, ne´ per la definizione e ne´ per la chiamata, e non ammette scambio formale di parametri con i sottoprogrammi, cioe´ cio´ che noi abbiamo scritto come (IN:parametri_ingresso;OUT:parametri_uscita).

3. Il BASIC non ammette variabili locali: tutte le variabili BASIC sono in comune fra tutti i programmi e sottoprogrammi nell´area di lavoro.

4. Abbiamo usato nomi di variabili e notazioni non ammesse in BASIC.

Passiamo in rassegna questi punti.

1. STRUTTURE DI CONTROLLO

Elenchiamo tutte le strutture di controllo pseudo-PASCAL, che abbiamo usato, con a fianco la traduzione in BASIC.

```
FOR K := N TO M                          1   FOR K = N TO M
DO BEGIN

        istruzioni;          <=>         istruzioni

END;                                     100 NEXT K
```

```
REPEAT                                   1   REM inizio loop

     istruzioni;             <=>         istruzioni

UNTIL (condizione);                      100 IF NOT (condizione) THEN  1
```

```
WHILE (condizione)                       1   IF NOT (condizione) THEN 200
DO BEGIN

       istruzioni;           <=>         istruzioni

END;                                     100 GOTO 1
                                         200 REM uscita loop
```

```
IF (condizione)                          1   IF NOT (condizione) THEN 100
THEN BEGIN
        istruzioni;          <=>         istruzioni

END;                                     100 REM fine IF ... THEN
```

```
IF (condizione)                          1   IF NOT (condizione) THEN 200
THEN BEGIN

        istruzioni 1;                        istruzioni 1

END;                          <=>
                                             199 GOTO 300
ELSE BEGIN                                   200 REM inizio ELSE

        istruzioni 2;                        istruzioni 2

END;
                                             300 REM fine IF ... THEN ... ELSE
```

Come avete visto BEGIN ed END non hanno una specifica traduzione, ma servono solo a delimitare blocchi di istruzioni di pari livello.
Il livello viene anche evidenziato graficamente scrivendo le istruzioni di ogni successivo livello sempre piu´ a destra.

2. SOTTOPROGRAMMI

Quelle che noi abbiamo chiamato procedure e sottoprocedure diventeranno tutte sottoprogrammi o subroutines, in quanto il BASIC non ammette la definizione di moduli o procedure ma solo di un programma principale e vari sottoprogrammi.
Il programma principale sara´ il vostro programma applicativo che richiamera´, quando necessario, i sottoprogrammi che realizzano le funzioni di traslazione e rotazione del robot.

Le procedure che abbiamo usato venivano chiamate con un nome, seguito dai parametri in ingresso ed in uscita.

Per il nome non c´e´ problema: in BASIC sostituiremo al nome il numero di linea e la chiamata della procedura sara´ "GOSUB numero-linea".

Ci sono piu´ problemi, e varie possibili soluzioni, per il passaggio di parametri.
In BASIC non e´ previsto uno scambio formale di parametri fra programma chiamante e sottoprogramma chiamato.
Siccome in BASIC non ci sono variabili locali (cioe´ che "esistono" solo per la procedura in cui sono dichiarate ed usate) lo scambio di parametri lo faremo attraverso delle variabili comuni: in ingresso con delle variabili che il programma chiamante scrive e la subroutine poi legge; in uscita con delle variabili che la subroutine scrive e poi il programma chiamante legge.

Per chiarezza vi consigliamo di riservare dei nomi di variabili ad ogni singola subroutine, anche se questo implica maggior spreco di memoria e, nel BASIC interpretato, anche di tempi di esecuzione.

Facciamo un esempio che chiarisca la traduzione.

Supponiamo di avere questa procedura:

```
BEGIN
    acc% := 0;
    vel% := 10;
    spazio% := 1000;
    tempo% := spazio% / vel%;
    tempo_fine% := 4;
    avanti% := 0;

    produci_poli(IN:avanti%,acc%,vel%,tempo_fine%,tempo%,posizione%;
                 OUT:polinomio$);

    stampa polinomio$;
END;
```

E la sottoprocedura produci_poli sia quella vista nel paragrafo C.5.2:

```
produci_poli(IN:verso%,A%,V%,TF%,TM%,PF%;OUT:poli$);

BEGIN
    indietro% := 1;
    PF_high% := INT( PF% / 4096 );
    PF_low% := INT( PF% - (PF_high% * 4096));

    IF ( verso% = indietro% )
    THEN BEGIN
             A% := A% OR 240;
             V% := V% OR 240;
             PF_high% := PF_high% OR 240;
    END;

    poli$ := CHR$(A%) + CHR$(V%) + CHR$(TF%) + CHR$(TM%)
             + CHR$(PF_low%) + CHR$(PF_high%);

    OUTPUT: poli$;
END;
```

La traduzione in BASIC sara´:

```
10  ACC% = 0
20  VEL% = 10
30  SPAZIO% = 1000
40  TEMPO% = SPAZIO% / VEL%
50  FINE% = 4
60  AVANTI% = 0
70  REM -------------------- SCAMBIO PARAMETRI INGRESSO
80  VERSO% = AVANTI%
90  A% = ACC%
100 V% = VEL%
110 TF% = FINE%
120 TM% = TEMPO%
130 PF% = SPAZIO%
140 GOSUB 1000 : REM --------CHIAMATA PRODUCI-POLI
150 PARAMETRI$ = POLI$ : REM SCAMBIO PARAMETRI USCITA
160 PRINT POLI$
170 END
1000 REM --------------------PRODUCI-POLI
1010 INDIETRO% = 1
1020 PH% = INT( PF% / 4096 )
1030 PL% = INT( PF% - (PH% * 4096))
1040 IF NOT ( VERSO% = INDIETRO% ) THEN 1100
1050 A% = A% OR 240
1060 V% = V% OR 240
1070 PH% = PH% OR 240
1100 POLI$ = CHR$(A%) + CHR$(V%) + CHR$(TF%) + CHR$(TM%) + CHR$(PL%)
             + CHR$(PH%)
1110 RETURN
```

Come possiamo vedere la pseudo-istruzione OUTPUT non viene tradotta nella subroutine, ma nel programma chiamante, e causa semplicemente la copiatura del valore della variabile di uscita, che noi abbiamo riservato alla subroutine, in una variabile che noi abbiamo riservato al programma chiamante.
Naturalmente voi potrete trovare altri modi, piu´ consoni al vostro modo di affrontare i problemi, per risolvere lo scambio con chiarezza e senza sovrascrittura non voluta di variabili.

3. VARIABILI LOCALI

In realta´ la cosa l´abbiamo gia´ vista al punto precedente.
La soluzione vista sopra implica pero´ la creazione di tante variabili, riservate alle singole subroutine.

Quando si puo´ evitare di creare nuove variabili in ogni subroutine?

Si puo´ usare un nome di variabile gia´ esistente solo se si e´ sicuri che tutte le routine che chiamano quella subroutine non abbiano piu´ bisogno, dopo l´esecuzione della subroutine, del valore contenuto in quella variabile prima dell´esecuzione della subroutine stessa.

4. NOTAZIONI

La notazione usata nel nostro pseudo-linguaggio e´ gia´ molto simile al BASIC; le uniche differenze sono:

- l´operatore di assegnazione ":=" si traduce in BASIC con "=".

- le righe di istruzione BASIC non terminano con ";".
  Occorre semplicemente eliminarlo.

- Il carattere "_" si puo´ inserire nel nome delle variabili BASIC.
  Per esempio nel CBM 64 o 128 si usa il simbolo che si ottiene premendo insieme i tasti "C=" e "@".
  Ricordate pero´ che non tutti i caratteri del nome delle variabili sono significativi in BASIC: per esempio nel CBM 64 o 128 solo i primi due caratteri sono significativi, e dunque due variabili che inizino con gli stessi due caratteri saranno uguali per l´interprete BASIC.

- la "stringa-vuota$" del nostro pseudo-linguaggio si traduce in BASIC con due doppi apici ("").

E.2: MODDY BAS: un BASIC per MODULUS

MODDY BAS e´ un interprete BASIC esteso che permette il controllo ad alto livello delle funzioni di MODULUS.

Anche MODDY BAS, come MODULUS, e´ modulare e cresce.

MODDY BAS e´ l´equivalente software di quello che e´ la spicchioteca per la parte hardware: nasce con una struttura base, ma puo´ ospitare molti "spicchi-funzione" software.
Per ogni funzione hardware che aggiungete a MODULUS e´ disponibile un equivalente spicchio-funzione software da aggiungere a MODDY BAS.

L´interprete BASIC MODDY BAS e´ fornito per Commodore 64 o 128 ed e´ perfettamente integrato con l´ambiente BASIC Commodore.
La struttura BASIC Commodore viene completamente mantenuta ed anche valori dei parametri non accettabili o errori nella sintassi dei comandi per MODULUS provocheranno l´apparizione dei consueti messaggi di errore del Commodore.

Le estensioni al BASIC Commodore per MODULUS in configurazione sola BASE sono:

- FWD dist,vel

  Questo comando fara´ muovere MODULUS lungo una linea retta per una distanza <dist> ad una velocita´ <vel>.
  Nel caso che la velocita´ non venga specificata, si assume la velocita´ minima consentita.

- MOVE velsx,temposx,veldx,tempodx

  Questo comando azionera´ i due motori (sinstro e destro) delle ruote di MODULUS alle velocita´ <velsx> e <veldx> per i rispettivi tempi <temposx> e <tempodx>.
  Nessun parametro puo´ essere omesso.
  Questo comando permette di effettuare movimenti molto generici (curve, spezzate, ...).

- ROT angl,vel

  Questo comando fara´ compiere a MODULUS una rotazione attorno al proprio asse di un angolo <angl> alla velocita´ <vel>.
  Valori positivi di <angl> causeranno una rotazione oraria e valori negativi una rotazione antioraria.
  Se la velocita´ viene omessa si assume la minima velocita´ consentita.

- REPORT

Questo comando chiedera´ a MODULUS l´invio del messaggio di stato BASE.
Tale messaggio verra´ conservato nella variabile riservata st.

- HALT

Questo comando provoca l´arresto immediato dei motori della BASE MODULUS.

- ROTP angl,vel

Questo comando e´ analogo a ROT, salvo che la rotazione di MODULUS avverra´ attorno alla posizione della penna del PLOTTER-DRAWING MECHANISM invece che attorno all´asse del robot.
Cio´ permette di tracciare angoli in fase di disegno.

- PENUP/PENDOWN

Questi comandi provocano l´alzamento o l´abbassamento della punta della penna-plotter oppure lo spegnimento o l´accensione dell´aspirapolvere.

Tutti i dettagli di MODDY BAS sono contenuti nel manuale di riferimento che SIRIUS commercializza insieme al package MODDY BAS.

Potrete sistemare tale fascicolo in questo manuale come paragrafo E.3 .

F HARDWARE

F.1 Descrizione cavi e connettori

F.1: Descrizione cavi e connettori

In questo paragrafo passeremo in rassegna prima tutti i connettori presenti sulla BASE MODULUS, sulla R.F.-COMMAND KEYBOARD e sul POWER SUPPLY & BATTERY CHARGER.
Seguira´ la descrizione dei cavi: il DATA & POWER CABLE, il cavo RS 232 standard fra personal/home computer e RF-CK, il cavo seriale Commodore fra CBM 64/128 e RF-CK.

CONNETTORI

1. BASE MODULUS

Abbiamo 4 connettori presenti sulla BASE MODULUS: PS & BC, I/O SERIALE, AUX, sonda umidita´.

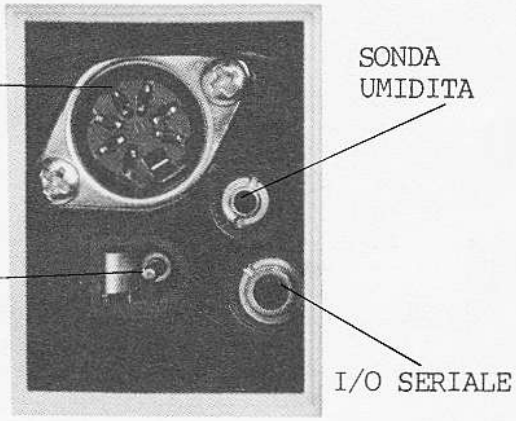


fig f.1.1

Il connettore PS & BC e´ una presa NUVAL 14011 per spina NUVAL 13080, o equivalente, da 9,5 mm di lunghezza, 5,5 mm di diametro esterno e 2 mm di diametro interno.

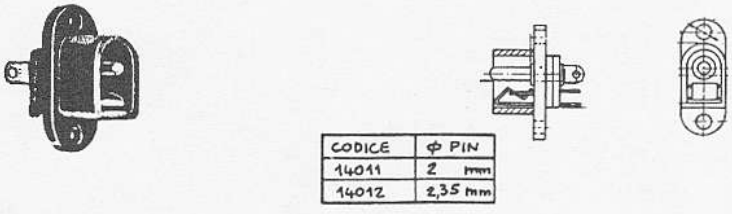


fig f.1.2

Nel caso di CABLE LINKED BASE (BS 101) non ci sono batterie a bordo e la presa consente l´alimentazione della BASE con il PS & BC (PS = Power Supply).
Al connettore occorre portare una tensione di 24 volt in corrente continua, con 0,5 Ampere di assorbimento medio.
Il negativo va connesso al contatto interno della spina.

Nel caso della RF LINKED BASE (BS 101/RF) sono presenti batterie ricaricabili (stagne al piombo) a bordo e la presa consente la loro ricarica.
Al connettore occorre portare una tensione di 32 volt in corrente alternata a vuoto, che scendono a 24 volt sotto carico di 0,6 Ampere.

ATTENZIONE: la corrente erogata NON DEVE superare gli 0,6 Ampere a 24 volt, pena l´accorciamento drastico della vita delle batterie.

- - -

Il connettore "I/O SERIALE" della BASE e´ una presa NUVAL 14099, o equivalente, per spinotto jack a 3 contatti da 3,5 mm di diametro (tipo jack stereo piccolo).

fig f.1.3

Al contatto esterno della spina va connesso il riferimento comune di massa GND.
Al contatto in punta corrisponde il segnale RXD della BASE, che andra´ connesso al segnale Sout (TXD) della porta USER I/O.
Al rimanente contatto corrisponde il segnale TXD della BASE, che andra´ connesso al segnale Sin (RXD) della porta USER I/O.
Ricordiamo che questi segnali sono di livello diverso e logicamente invertiti rispetto ai corrispondenti segnali RS 232 standard.
Per ulteriori dettagli leggete il paragrafo D.2 .

- - -

Il connettore AUX e´ una presa DIN a 7 poli, tipo NUVAL 5060 PD, o equivalente.

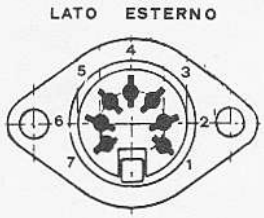


fig f.1.4

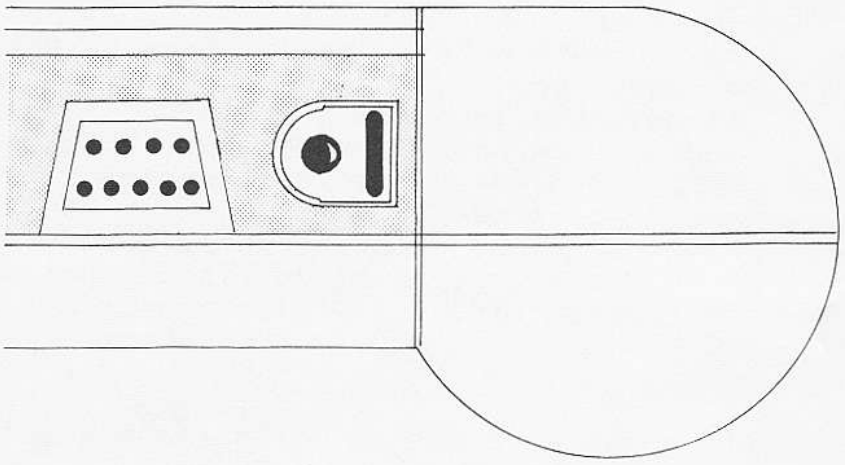
Su questa presa sono disponibili 24 volt in corrente continua e max 1,5 Ampere (con BASE alimentata da batterie) per alimentare l´aspirapolvere o l´abbassapenna del plotter-drawing mechanism.
ATTENZIONE: l´aspirapolvere puo´ funzionare SOLO se connesso alla BASE alimentata con batterie (BS 101/RF).
Inoltre abbiamo un ingresso per il segnale di "penna giu´" proveniente dall´alzapenna.

I contatti sono cosi distribuiti:

- pin 1,2 segnale di "penna abbassata" (1) e riferimento (2)
- pin 3,4 negativo 24 volt
- pin 5,6 positivo 24 volt
- pin 7 non connesso

Il connettore per la sonda di umidita´ e´ una presa NUVAL 14059, o equivalente, per spinotto jack a 2 contatti da 2 mm di diametro.
A questa presa andra´ connessa la sonda del rivelatore di umidita´ contenuto nello spicchio "HOME & SECURITY": riviamo al relativo manuale per ulteriori informazioni.

2. RF-COMMAND KEYBOARD



Abbiamo 2 connettori sulla RF-CK: alimentazione esterna e interfaccia seriale verso il computer di controllo.

fig f.1.5

Il connettore di alimentazione esterna e´ una presa NUVAL 14012 per spina NUVAL 13081, o equivalente, di 9,5 mm di lunghezza, 5,5 mm di diametro esterno e 2,5 mm di diametro interno.

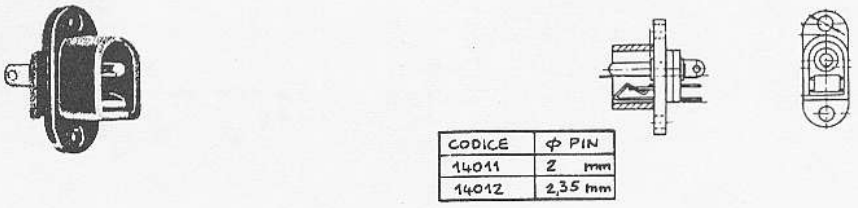


fig f.1.6

Al connettore occorre fornire una tensione di 12 volt in corrente continua, con 0,2 Ampere di assorbimento.
Il negativo va connesso al contatto interno della spina.

- - -

Il connettore per interfaccia seriale (RS 232 e Commodore) e´ del tipo a vaschetta 9 poli (tipo "Cannon" o DB 9).

fig f.1.7

I segnali sono cosi´ distribuiti:

- pin 1 GND : collegare a GND RS 232/Commodore
- pin 2 RTS (DCE) - RS 232 -: collegare a RTS del computer (DTE)
- pin 3 TXD (DCE) - RS 232 -: collegare a TXD del computer (DTE)
- pin 4 RXD (DCE) - RS 232 -: collegare a RXD del computer (DTE)
- pin 5 CTS (DCE) - RS 232 -: collegare a CTS del computer (DTE)
- pin 6 GND : come pin 1
- pin 7 RX (DCE) - Commodore -: collegare a Sin Commodore (DTE)
- pin 8 TX (DCE) - Commodore -: collegare a Sout Commodore (DTE)
- pin 9 GND : come pin 1

Per informazioni su segnali e nomi vedere paragrafo D.3 .

3. POWER SUPPLY & BATTERY CHARGER - - -

Abbiamo 3 connettori/cavi presenti sul PS & BC: presa alimentazione BASE, cavo/spina caricabatterie, cavo/spina rete 220 volt.

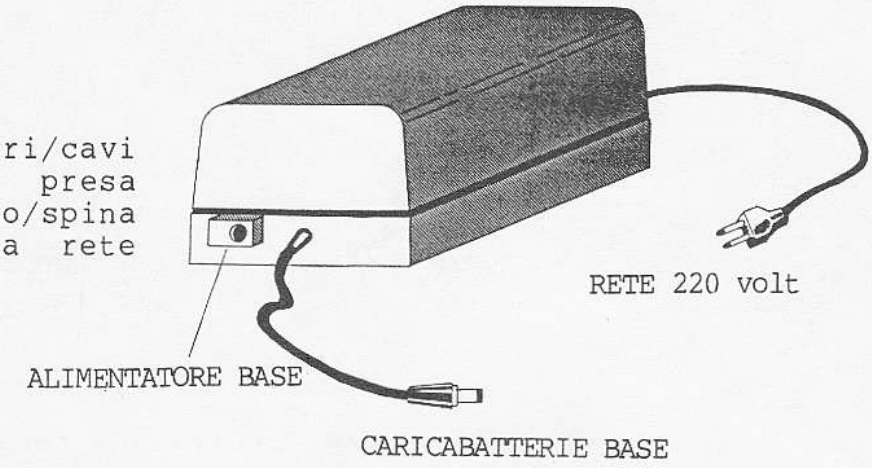


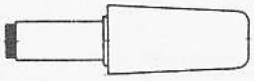
fig f.1.8

Il connettore alimentazione BASE e' una presa NUVAL 14091, o equivalente, per spina jack da 3,5 mm di diametro esterno.

fig f.1.9

Viene usata nel caso di CABLE LINKED BASE (BS 101), che non ha batterie a bordo, per l'alimentazione della BASE con il PS & BC (PS = Power Supply).
Al connettore e' presente una tensione di 24 volt in corrente continua, con 0,5 Ampere di erogazione.
Il negativo va connesso al contatto esterno della spina, il positivo in punta.

Il cavo/connettore di caricabatterie e´ un cavo lungo 2 metri, che esce direttamente dalla scatola del PS & BC con in testa una spina NUVAL 13082, o equivalente, da 14,5 mm di lunghezza, 5,5 mm di diametro esterno e 2,5 mm di diametro interno.



⌀ EST.	⌀ INT.	CODICE
5,5	2,1	13082
5,5	2,5	13083

fig f.1.10

Viene usato nel caso della RF LINKED BASE (BS 101/RF) per ricaricare le batterie (stagne al piombo) a bordo con il PS & BC (BC = Battery Charger).
Al connettore e´ presente una tensione di 32 volt in corrente alternata a vuoto, che scende a 24 volt sotto carico di 0,6 Ampere.

\- - -

Il cavo/spina di rete e´ un normale cordone di alimentazione con spina a 2 contatti da inserire in una presa di rete a 220 volt alternati.

CAVI

1. DATA & POWER CABLE - - -

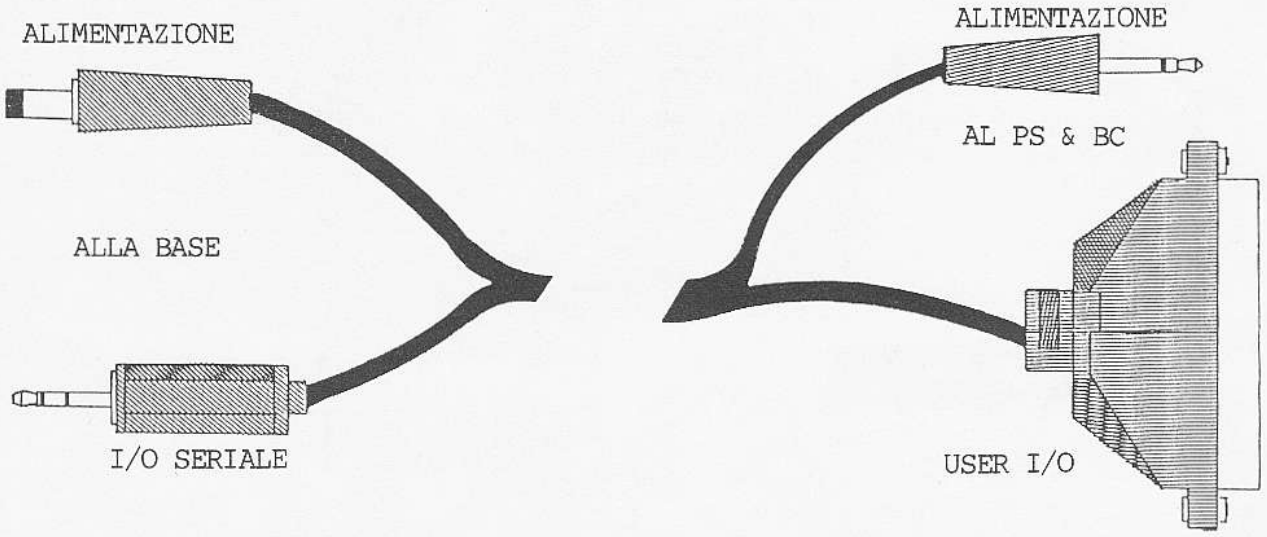


fig f.1.11 ALLA PORTA USER I/O COMMODORE

Viene usato nel caso di CABLE LINKED BASE (BS 101), che non ha ne´ batterie ne´ modem a radio frequenza a bordo, per l´alimentazione e la comunicazione con il computer di controllo (CBM 64 o 128).

Il cavo e´ lungo circa 5 metri ed e´ composto da 5 fili e 4 connettori.

Due fili e due connettori (una spinetta tipo NUVAL 13080, o equivalente, da 9,5 mm di lunghezza 5,5 mm di diametro esterno e 2,1 mm di diametro interno e un jack da 3,5 mm di diametro con due contatti, tipo NUVAL 13064 o equivalente) sono riservati per portare l´alimentazione dal PS & BC alla BASE.

Sono collegati insieme il contatto in punta del jack con quello esterno del 13080 e il corpo del jack con quello interno del 13080.

Tre fili e due connettori, uno a jack da 3,5 mm (tipo stereo: NUVAL 13063, o equivalente) e l´altro a 12+12 poli a pettine con passo 3,96 mm, convogliano i segnali di comunicazione fra BASE e CBM 64/128.

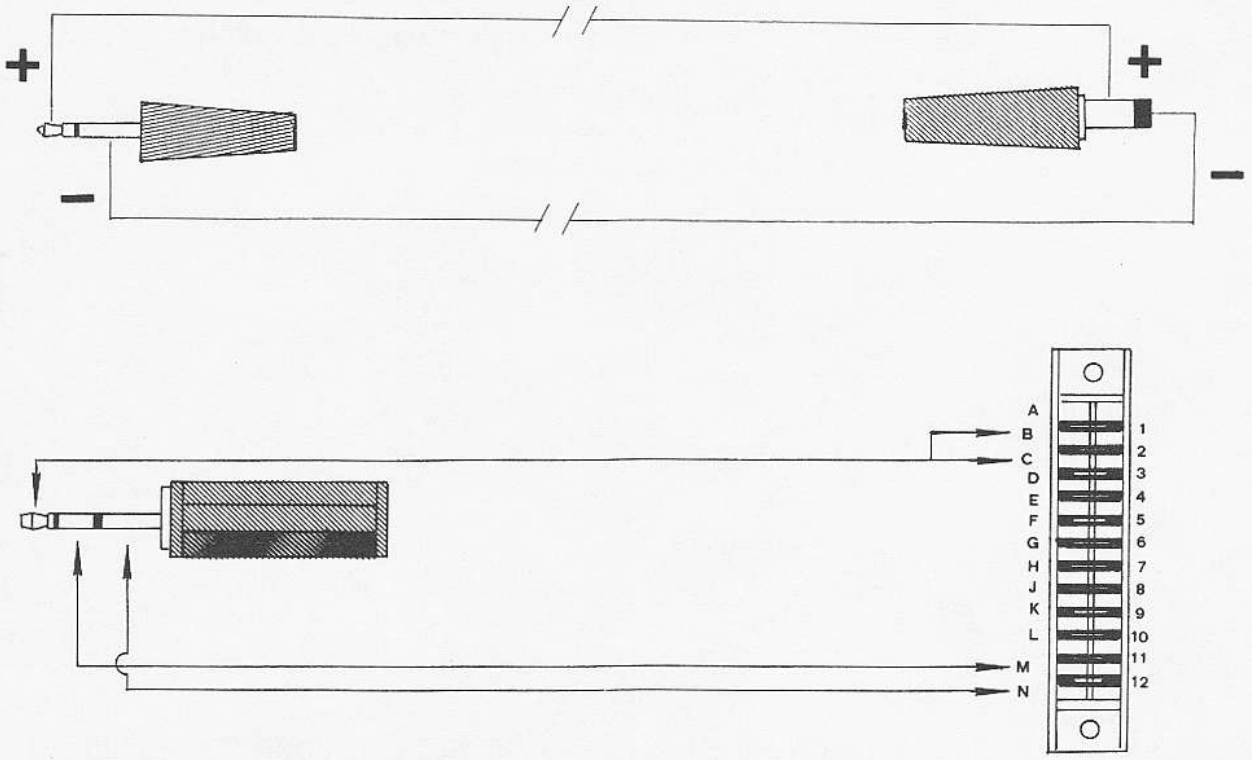


fig f.1.12

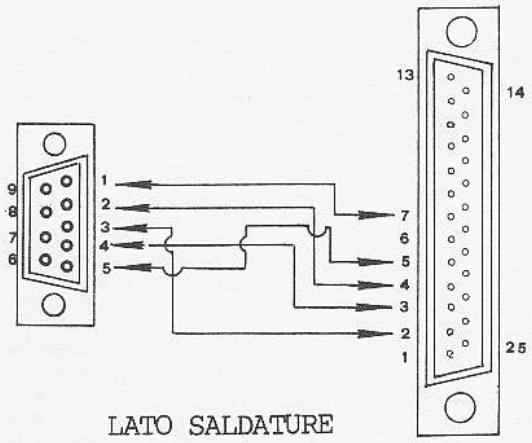
Il contatto esterno del jack va connesso al pin N del connettore a pettine (GND <--> GND).
Il contatto in punta del jack va connesso al pin M del connettore a pettine (RXD <-- Sout/TXD).
Il rimanente contatto del jack va connesso ai pin B e C del connettore a pettine (TXD --> Sin/RXD).
Quando il connettore a pettine e´ inserito nella porta USER I/O del CBM, tutti i pin del connettore a pettine interessati al collegamento si trovano nella fila inferiore.
Per ulteriori notizie leggete il paragrafo D.2 .

2. CAVO COMPUTER <--> RF-CK - - -

Questo cavo connette l´interfaccia seriale RS 232 del vostro home/personal computer a quella dell´RF-CK.
Il cavo e´ composto da 5 fili e 2 connettori.
Un connettore e´ un maschio a vaschetta 9 poli (tipo "Cannon" o DB 9).
L´altro e´ in genere a vaschetta a 25 poli (sempre tipo "Cannon" o DB 25) ed e´ maschio o femmina a seconda che la presa RS 232 a pannello del vostro computer sia femmina o maschio.
Puo´ anche essere che il vostro computer abbia un diverso connettore per l´interfaccia RS 232: in tal caso consultate il relativo manuale per localizzare i segnali necessari (vedi pagina seguente).

La numerazione dei pin che diamo si basa su un connettore a vaschetta da 25 poli.

CON HANDSHAKE (9600 O 300 BD)



SENZA HANDSHAKE (SOLO 300 BD)

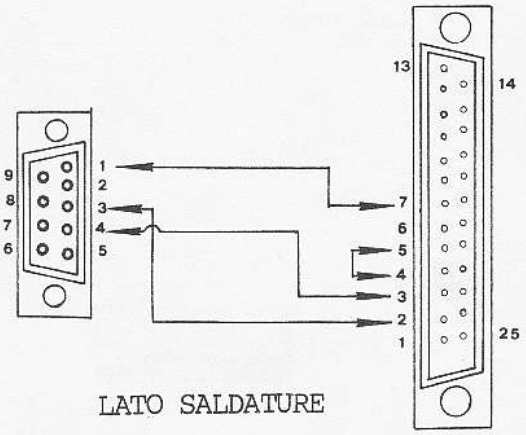


fig f.1.13

Collegare:
- il pin 1 del 9 poli al pin 7 del 25 poli (SGND <--> SGND).
- il pin 2 del 9 poli al pin 4 del 25 poli (RTS<dce> <-- RTS<dte>).
- il pin 3 del 9 poli al pin 2 del 25 poli (TXD<dce> <-- TXD<dte>).
- il pin 4 del 9 poli al pin 3 del 25 poli (RXD<dce> --> RXD<dte>).
- il pin 5 del 9 poli al pin 5 del 25 poli (CTS<dce> --> CTS<dte>).

Questo cavo e' previsto per handshake su RTS/CTS: se intendete usare l'interfaccia senza handshake (questo implichera' usare la velocita' di 300 baud) bastera' realizzare i seguenti collegamenti:

- il pin 1 del 9 poli al pin 7 del 25 poli (SGND <--> SGND).
- il pin 4 del 9 poli al pin 3 del 25 poli (RXD<dce> --> RXD<dte>).
- il pin 3 del 9 poli al pin 2 del 25 poli (TXD<dce> <-- TXD<dte>).

Se volete avere ulteriori informazioni consultate il paragrafo D.3 .

3. CAVO CBM 64/128 <--> RF-CK - - -

Questo cavo connette l´interfaccia seriale Commodore a quella dell´RF-CK.
Il cavo e´ composto da 3 fili e 2 connettori.
Un connettore e´ un maschio a vaschetta 9 poli (tipo "Cannon" o DB 9).
L´altro e´ a pettine da 12 + 12 poli, passo 3,96 mm, che andra´ inserito nella porta USER I/O del Commodore.

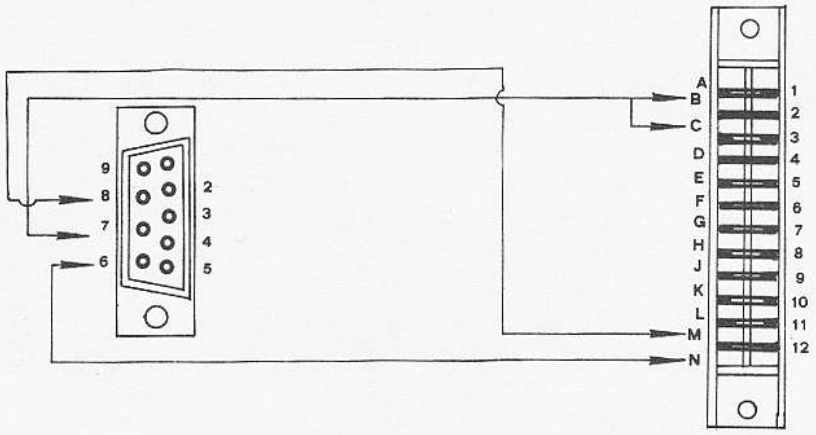


fig f.1.14

Collegare:
- il pin 6 del 9 poli al pin N del 12+12
  (SGND <--> SGND).
- il pin 8 del 9 poli al pin M del 12+12
  (TX<dce> <-- Sout/TX<dte>).
- il pin 7 del 9 poli ai pin B,C del 12+12
  (RX<dce> --> Sin/RX<dte>).

Per ulteriori informazioni consultate il paragrafo D.3 .

G BIBLIOGRAFIA

G.1: Per saperne di piu´

I robot sono uno strumento molto fertile per ragionare: da una parte ci pongono da problemi estremamente pratici e dall´altra, in quanto macchine, ci costringono ad usare strumenti teorici per risolverli.

Insomma, la robotica e´ un affollatissimo campo di incrocio di discipline scientifiche (e non solo!) che vengono costrette a misurarsi con la prova piu´ terribile: l´azione in uno spazio reale.

Mentre il software vive nel "soffice" mondo del computer, mentre le teorie possono vivere nel loro limbo mentale, il robot costringe a confrontarsi con il nostro mondo reale.

Per saperne di piu´ occorre interessarsi alla fisica del movimento, dalla cinematica alla dinamica dei corpi rigidi: abbiamo solo assaggiato in questo manuale la programmazione del moto di MODULUS, le accelerazioni, le frenate, gli slittamenti, le traiettorie,... .

E poi la geometria in 2 o 3 dimensioni: per esempio, come tenere il punto nello spazio di movimento, come arrivare ad afferrare un oggetto quando il nostro MODULUS avra´ un braccio e una mano,... .

Poi potreste interessarvi dei sistemi di regolazione o di controllo, applicazioni pratiche della prima scoperta della cibernetica: per esempio, gli anelli di regolazione dei motori, che usano tecniche di feedback o retroazione.

Un´altro campo che avete avuto quasi sempre sotto gli occhi nel "viaggio" attraverso questo manuale: la telematica e la trasmissione dei dati.

Il sistema nervoso di MODULUS e´ una rete di trasmissione dati.
Il nostro controllo su MODULUS dipende essenzialmente dalla comunicazione: in un certo senso non e´ poi vero che lo scopo della comunicazione e´ il controllo?

Poi le strutture con cui formalizzare la parte algoritmica del nostro pensiero: noi abbiamo proposto uno pseudo-linguaggio per computer che fa uso di strutture di controllo che si potrebbero studiare meglio e in ogni caso abbiamo progettato un linguaggio per il movimento.

Cosa si puo´ fare con un linguaggio per il movimento?
La domanda non e´ stupida: chi conosce non superficialmente il mondo della tartaruga LOGO sa che il ragionamento spaziale puo´ essere terribilmente piu´ potente di quello verbale e simbolico, e non si esaurisce solo nel disegnare un quadrato.

Poi il mondo dei sensori e delle interfacce simil-uomo, o human-like.

Il progetto MODULUS mette a disposizione molti sensori: di urto, un "naso", un "orecchio direzionale", un sonar, un "occhio infrarosso"; e anche sofisticati sistemi di interazione con gli umani: voce, manipolazione spaziale, ...

E infine l´intelligenza artificiale: un campo preteso da molti ma, in realta´, saggiato da pochi.

Un robot deve misurarsi con la percezione dell´ambiente, con l´apprendimento, con il ragionamento su oggetti nello spazio, con la navigazione in un ambiente reale.

La robotica e´ il "braccio" dell´intelligenza artificiale nella realta´.

G.2: Robotica e intelligenza artificiale

Approfondiamo per un attimo il legame tra robotica e intelligenza artificiale.

Come l´ha felicemente definita Micheal Brady, del lab. of Artificial Intelligence del M.I.T., la robotica e´ la connessione intelligente tra la percezione e l´azione.

E´ la parola "intelligente" che trasforma la sensazione in percezione e che puo´ produrre una adeguata azione del robot sull´ambiente.

I robot industriali delle catene di montaggio stanno cominciando ora a porsi realisticamente il problema dell´intelligenza.
Ancora oggi, spesso, sono macchine veloci, soprattutto instancabili e precise, ma incredibilmente stupide ed incapaci di percepire e ragionare sul loro pur ridottissimo mondo.

Un robot come MODULUS puo´ essere un ottimo banco di prova per sviluppare software con delle capacita´ di apprendimento e di organizzazione della conoscenza.

Navigare in una stanza e´ un problema tutt´altro che banale!
Provate a rappresentare la struttura della stanza in cui opera MODULUS: provate a rappresentare il suo movimento in questo ambiente, anche solo bidimensionalmente.
Provate ad immaginare la vostra stanza divisa a scacchiera in tanti piccoli quadrati in cui individuare muri e oggetti.
Pensate a come far esplorare questo mondo a scacchiera a MODULUS.

Di pensiero in pensiero siamo venuti a contatto con alcuni dei piu´ affascinanti quesiti sulla rappresentazione mentale dello spazio.

Con la BASE abbiamo solo dei sensori di urto, ma appena MODULUS cresce puo´ disporre di un sonar con cui "guardarsi" intorno e determinare distanze e anche alcuni dispositivi per fare il punto nell´ambiente, aiutandosi con dei "radiofari" all´infrarosso.

Perche´ non pensare ad un piccolo sistema esperto che, inglobando conoscenze euristiche sul funzionamento del mondo-stanza in cui si muove il robot, non si integri con la percezione sensoriale per riconoscere alcune situazioni che incontra nella navigazione?

E´ inutile nascondere che nessuno di questi quesiti e´ facile e completamente risolvibile, ma ogni nuova idea sara´ immediatamente verificabile facendo muovere MODULUS nello spazio-mondo reale.

G.3: Letture consigliate

Diamo qui una bibliografia per cominciare a "saperne di piu´".
Non c´e´ nessuna pretesa di completezza, ma solo quella di suggerire dei buoni mezzi per apprendere a sfruttare meglio MODULUS e per arricchire le vostre conoscenze.

I libri sono organizzati per argomenti, grosso modo nello stesso ordine in cui li abbiamo individuati nel paragrafo G.1.

Per ogni argomento i libri sono ordinati per utilita´, partendo dal presupposto che il lettore sia a digiuno della materia.

I libri sono sia in italiano che in inglese: i testi in inglese sono stati indicati solo dove la "letteratura" italiana e´ carente.
I testi in lingua inglese sono marcati con un "@" davanti al titolo; in ogni caso i testi esteri citati sono reperibili normalmente nelle librerie specializzate in informatica ed elettronica.

I prezzi riportati sono indicativi e SIRIUS non si assume nessuna responsabilita´ su eventuali discordanze.

ROBOTICA

1. @ Introduction to Robotics: Mechanics & Control
   J.J. Craig — ed. ADDISON WESLEY — £ 78000

2. @ Introduction to Robotics
   Critchlow — ed. MACMILLAN — £ 56000

3. Robot
   T.Logsdon — ed. SUPERNOVA — £ 19500

4. Robot
   Gini/Gini — ed. CLUP — £ 13000

5. Robotica
   Seva/Somalvico — ed. CLUP — £ 10000

FISICA DEL MOVIMENTO

6. Elementi di fisica per l´universita´ - vol 1: Meccanica
   Alonso/Finn ed. MASSON / ADDISON WESLEY £ 43000

7. La fisica di Berkeley: meccanica
   Kittel/Knight/Ruderman ed. ZANICHELLI £ 41500

8. @ Lectures on Physics
   Feynman/Leighton/Sands ed. ADDISON WESLEY £ 46000

9. Fisica vol. 1
   Resnik/Halliday ed. AMBROSIANA £ 38000

CONTROLLO E REGOLAZIONE

10. Sistemi di controllo
    N.M. Morris ed. HOEPLI £ 20000

11. Il controllo automatico dei sistemi
    Schmitt/Farwell ed. JACKSON £ 29500

12. Sistemi
    Capezzuto/Gianni ed. HOEPLI £ 20000

13. Controlli automatici
    G.Pacquino ed. CLUP £ 16000

PROTOCOLLI E TELEMATICA

14. Trasmissione dati: dispositivi standard e protocolli
    G.Saccardi ed. JACKSON £ 23000

15. L´interfaccia RS 232
    M.D.Seyer ed. MASSON £ 30000

16.@ A Guide to Data Communication
    V.P.Clare ed. CASTLE HOUSE PUBLICATIONS £ 19000

17. Telematica: trasporto e trattamento dell´informazione
    Macchi/Guilbert ed. TECNICHE NUOVE £ 42000

18. Telematica: architetture, protocolli e servizi
    G.Le Moli ed. ISEDI / A.MONDADORI £ 32000

PROGRAMMAZIONE

19. Introduzione alla programmazione strutturata
Lanzarone/Maiocchi/Polillo ed FRANCO ANGELI £ 20000

20. Soluzione di problemi con Pascal
Kenneth/Bowles ed. JACKSON £ 28000

21. Programmare in Pascal
David/Deschamps ed. JACKSON £ 16000

"TARTARUGHE" INFORMATICHE

22. Mindstorms
Papert ed. EMME EDIZIONI £ 20000

23. La geometria della tartaruga
Di Sessa/Abelson ed. FRANCO MUZZIO £ 48000

24. LOGO: manuale per Commodore 64
Grammer/Goldenberg/Klotz ed. COMMODORE/ECKENWILLER

25. LOGO: ali per la mente
Reggini ed. MONDADORI £ 20000

HARDWARE

26. Microelaboratori: fondamenti
Cavalcoli/Leone/Offeli ed. CLUP £ 20000

27. Microeleboratori: note di hardware
Baccolini/Offeli ed. CLUP £ 20000

INTELLIGENZA ARTIFICIALE

28. Verso l´intelligenza artificiale
Bundy/Burstall/Weir/Young ed. MONDADORI £ 28000

29. Intelligenza Artificiale
E.Rich ed. MC GRAW-HILL £ 39000

30.@ Artificial Intelligence
P.H.Winston ed. ADDISON WESLEY £ 76000

31. Sistemi esperti per il vostro computer
C.Naylor ed. TECNICHE NUOVE £ 25000

32. Costruire un sistema esperto
A.Mazzetti ed. FRANCO MUZZIO £ 38000

33. Il Basic per l´intelligenza artificiale
T.Hartnell ed. MONDADORI £ 28000

e da consultare nelle biblioteche specializzate

34.@ The Handbook of Artificial Intelligence
Barr/Feigenbaum/Cohen ed. WILLIAM KAUFMANN/ PITMAN

Milano-Fiori Palazzo F2 - 20094 Assago (MI) Italy - Tel. (02) 8245321