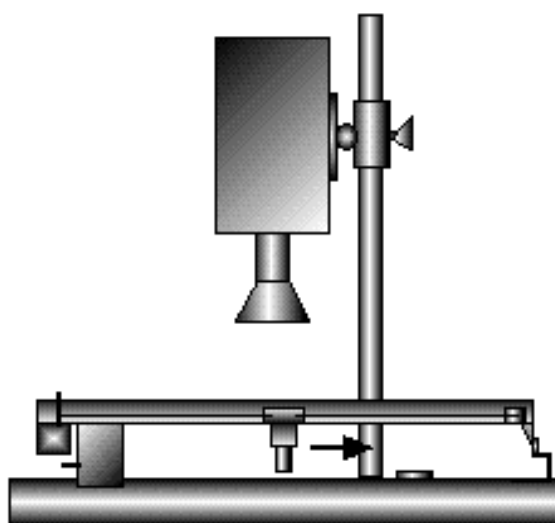


Istituto Professionale di Stato per
l'Industria e l'Artigianato
MORETTO
Via Luigi Apollonio, 21 BRESCIA

Automatismo Computerizzato e Monitorizzato per il Trasporto di Oggetti



A cura di:

DAMINELLI MICHELE
FILIPPINI MAURIZIO
PIACENTINI DEVIS

della classe 5BI a.s. 1995/96
corso per Tecnici delle Industrie Elettriche ed Elettroniche

PRESENTAZIONE

Nella società moderna, ed in particolar modo nel settore industriale, i sistemi automatizzati si sono sempre più diffusi, poiché grazie alle loro caratteristiche di velocità affidabilità e precisione, permettono un maggior livello qualitativo e quantitativo della produzione, migliorando il lavoro del personale addetto.

Il progetto da noi realizzato, non fa altro che riprodurre uno dei più diffusi tipi di automatismi: un “**ROBOT**” che riconosce automaticamente un determinato oggetto e che, dopo averlo prelevato, lo trasporta in un luogo prefissato dove sarà poi rilasciato.

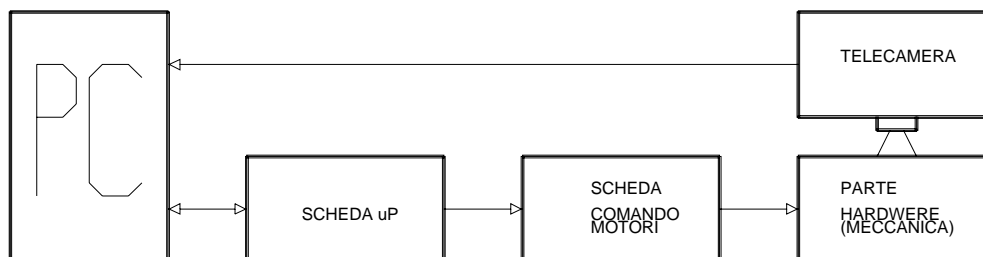
Questo nostro dispositivo, sebbene di impostazione didattica, trova spesso applicazione negli apparati produttivi, infatti può essere utilizzato per i più svariati scopi, come ad esempio nel controllo della qualità (dove il prodotto non ritenuto valido verrebbe scartato automaticamente), nel reparto spedizioni (dove, a seconda del tipo di prodotto e della sua destinazione sarebbe raggruppato in un determinato posto), oppure come un normalissimo mezzo di spostamento delle merci da un luogo ad un altro.

Nel nostro caso specifico, a sovrintendere le varie operazioni da compiere, vi è un personal computer, corredato un programma scritto in linguaggio **PASCAL**, che esegue il riconoscimento del determinato oggetto tramite una normale telecamera **VHS-C** interfacciata ad esso, e manda le istruzioni da compiere ad una scheda elettronica esterna che ha il compito di comandare la parte meccanica che effettua il prelievo, lo spostamento ed il rilascio dell'oggetto.

Tali spostamenti avvengono sulla superficie di un piano; il dispositivo di prelievo non è altro che un'elettrocalamita, di conseguenza l' oggetto dovrà essere obbligatoriamente di tipo ferromagnetico.

ARCHITETTURA DEL SISTEMA

Il sistema è strutturato secondo il seguente schema a blocchi:



Come si può notare, il blocco principe che sovrintende a tutto il sistema è il **PERSONAL COMPUTER** ed il programma in **PASCAL** in esso contenuto, in quanto qualsiasi dato viene mandato ad esso che lo elabora e lo fornisce agli altri blocchi facenti parte del sistema.

Anche la telecamera assume notevole importanza, infatti è affidata ad essa il compito di riconoscere la posizione dell' oggetto sul piano; ovviamente, essendo una normalissima telecamera **VHS-C** di tipo videoamatoriale, abbiamo dovuto munirci di una speciale scheda elettronica **VIDEOBLASTER** della **CREATIVE** per rendere possibile il collegamento tra essa ed il **PC**.

A questo punto, entrano in gioco tutte le altre parti del sistema, che hanno il compito di effettuare, anche fisicamente, i comandi imposti dal computer.

Il primo blocco che incontriamo è la scheda del microprocessore 6501, una sorta di microcomputer col compito di trasformare le informazioni provenienti dal **PC** in segnali adatti per far funzionare il successivo circuito di comando dei motori passo-passo; ciò è possibile tramite un programma scritto in **ASSEMBLY**, l' unico linguaggio riconosciuto dal μ P.

Guardando lo schema a blocchi, si può notare come, diversamente dagli altri collegamenti, quello tra il **PC** ed il μ P **6501** della **ROCKWELL**, sia di tipo "bidirezionale", cioè di andata e di ritorno, ciò è possibile proprio per il fatto che, come accennato prima, il 6501 non è altro che un computer in miniatura, in grado anche di "comunicare" con altri dispositivi; questo scambio di informazioni è reso possibile tramite un' interfaccia di comunicazione seriale denominato **RS 232**.

Per quanto riguarda gli ultimi due blocchi, ad essi è affidata la realizzazione pratica di tutto il progetto; la scheda di comando dei motori riceve le istruzioni dal microprocessore a proposito del verso, della

velocità e della quantità del movimento. Invia il tutto sottoforma di impulsi elettrici ai motori, i quali, tramite un sistema di spostamento assimilabile a quello dei plotter, realizzano i cambiamenti di posizione sul piano dell' elettromagnete, cioè del nostro dispositivo di prelievo degli oggetti.

In particolare, ecco di seguito la spiegazione di ogni singolo blocco facente parte del sistema.

CENNI SUL COMPUTER

Il sistema si può dividere in due parti fondamentali: il computer e l'elettronica di comando (hardware esterno). Nel computer é naturalmente compreso anche il software scritto in Pascal.

IL PERSONAL COMPUTER (PC). Il computer viene sempre considerato nelle sue due parti fondamentali: l'hardware ed il software. Per hardware si intende la struttura fisica di un elaboratore, formata da tutto ciò che può essere "toccato con mano": parti meccaniche, schede elettriche, elettroniche. Nella terminologia anglosassone la parola "hardware" significa infatti "ferramenta" o "parte dura". Per software ("parte molle") si intende l'insieme dei programmi che consentono la gestione del sistema e la gestione di tutti i problemi che la macchina deve risolvere. Il termine "software" comprende inoltre tutti i linguaggi di programmazione ed i dati immagazzinati o gestiti da un sistema a microprocessore.

L'HARDWARE DEL PERSONAL COMPUTER. In un personal computer si possono distinguere le seguenti unità hardware che svolgono le seguenti funzioni:

1) Unità di ingresso o "input", che acquisiscono dati dall'esterno per immetterli nel processo di elaborazione. Queste unità possono essere costituite da schede elettroniche interne alla macchina ma comprendono anche tastiera, mouse, scanner e qualsiasi periferica che consenta l'acquisizione dati dall'esterno. Non fanno quindi parte di queste unità le memorie in generale.

2) Unità di uscita o "output", che svolgono la funzione di mandare all'esterno i dati elaborati dal sistema. Queste unità possono essere costituite da schede di output interne alla macchina ma comprendono anche il video, la stampante, il plotter, schede audio, ecc.

3) Unità centrale di elaborazione o C.P.U. (Central Process Unit). Questa unità é tipicamente la "mother board" che comprende microprocessore (il cuore del sistema), memorie RAM e ROM, dispositivi di interfaccia (I/O). La CPU é costituita da due elementi inscindibili: l'unità di controllo e l'unità aritmetico logica o A.L.U. Il "cervello" del sistema ha invece il compito di mettere in collegamento con tutti i dispositivi periferici e di memoria con i quali scambiare dati.

4) Unità interna di memoria. I chip di memoria presenti nella “mother board” sono di due tipi: memoria RAM (Random Access Memory) e ROM (Read Only Memory).

La RAM memorizza le istruzioni dei programmi da eseguire e i dati su cui le istruzioni operano; fornisce alla CPU e alla ALU le istruzioni da eseguire secondo l'ordine logico in cui devono operare; memorizza i dati intermedi e i risultati finali di un programma.

La RAM é una memoria volatile, cioè, in mancanza di alimentazione tutti i dati impressi su di essa vengono persi.

La ROM, memoria non volatile e di sola lettura, contiene i dati inseriti dal costruttore.

Nella memoria ROM si trova il programma che serve al computer per avviarsi all'accensione.

5) Unità esterna di memoria (memoria di massa). Si indicano con questo nome quelle componenti che permettono di memorizzare dati come il FLOPPY DISK e l'HD. L'evoluzione tecnologica ha fatto in modo che negli ultimi tempi siano state realizzate nuove memorie di massa: il disco rigido estraibile, che rimane comunque “non tascabile”, e i cosiddetti CD-rom, che sono di sola lettura ma di elevata capacità (600 Mb) e modesta velocità di accesso.

IL SOFTWARE DEL PERSONAL COMPUTER. Il software é la seconda componente fondamentale di un computer.

Il software é in mezzo con cui l'uomo comunica con la macchina e si può distinguere in:

- 1) Software applicativo.
- 2) Software di base.

Il software applicativo é costituito dall'insieme dei programmi con cui l'utente utilizza il computer.

Il software di base é l'insieme dei programmi senza i quali il computer non può funzionare.

Fanno parte del software di base:

- 1) Sistema Operativo (O.S.);
- 2) Programmi di utilità che integrano le funzioni del sistema operativo.

Il sistema operativo é un complesso di programmi sovrintendente alle elaborazioni del calcolatore.

Il sistema operativo ha due scopi fondamentali:

- A) Semplificazione delle operazioni di impiego del calcolatore;
- B) Sfruttamento ottimale delle potenzialità della macchina.

SOFTWARE DEL SISTEMA

Come é già stato detto, il compito del personal computer é quello di gestire tutto il sistema, ma ciò é possibile solo tramite un programma scritto in linguaggio **TURBO PASCAL** della **BORLAND**, essendo tale linguaggio dotato di librerie per gestire la grafica dello schermo.

Esso, é strutturato in 25 corti sottoprogrammi, gestiti da un programma principale. Il programma principale si apre con il comando **USES**, che serve per specificare le librerie utilizzate, nel nostro caso la libreria grafica (**GRAPH**) e quella per la gestione del mouse (chiamata appunto **MOUSE**)

Di seguito, troviamo la definizione delle costanti (**CONST**) e dalle variabili (**VAR**) del programma; da notare che all'interno delle costanti ci sono tutta una serie di **ARRAY**, che servono per definire le coordinate delle varie parti grafiche che compongono la schermata.

Dopodiché, inizia tutta la serie di sottoprogrammi, divisi tra funzioni (**FUNCTION**) e **PROCEDURE**; la differenza fondamentale tra i due metodi é che al primo è sempre associato un valore, mentre per la seconda non è necessario.

La prima funzione che troviamo é quella denominata **HEX_ASC**, che ha il compito di tradurre il valore dello spostamento e del checksum da **HEX** ad **ASCII**.

La funzione **RS-INPUT** serve invece per acquisire dall'ingresso della porta seriale **COM 2** del **PC** un carattere proveniente dal μ P (**ACK** e **NACK**).

Per quanto riguarda la funzione **RS-STRING-INPUT**, viene utilizzata per ricevere in ingresso una nuova stringa di caratteri provenienti dal microprocessore 6501.

La funzione successiva **CHECKSUM**, serve per calcolare la somma dei valori hex attribuiti ad ogni carattere della stringa.

Infine, le ultime due funzioni compongono la stringa (**FUNCTION CODA**) e verificano che il valore del **CHECKSUM** corrisponda a quello inviato dal microprocessore (Function **TEST-CHC**)

Da questo punto in poi inizia la lunga serie di **PROCEDURE**

La prime, **RS-INIT**, inizializza il canale seriale, realizzando così il collegamento tra i due elaboratori; dopodiché viene utilizzato dalle due successive procedure per restituire un carattere (**RS-OUTPUT**) e per restituire la stringa intera (**RS-STRING-OUTPUT**).

La procedura **SHOW-IMM**, ha il compito di prelevare dal file l'immagine in formato **BMP** e caricarla nella memoria video del computer.

Le successive procedure **M-ON**, **M-OFF**, **SPOSTAMENTO-X**, **SPOSTAMENTO-Y**, **ORIGINE**, **CONTENITORE** e **STOP**, realizzano la

composizione delle rispettive stringhe, cioè i rispettivi comandi da mandare al microprocessore; essi vengono abilitati dalla procedura successiva.

Infatti, la procedura **LEGGI MOUSE** serve proprio per riconoscere se il tasto del mouse è stato premuto in determinati punti dello schermo; se ciò è avvenuto, attiva i sottoprogrammi precedenti che permettono la comunicazione col microprocessore e quindi la realizzazione pratica del comando.

Le espressioni matematiche contenute in quest'ultimo sottoprogramma, servono proprio per determinare le aree dello schermo all'interno delle quali il programma è in grado di riconoscere se il puntatore è stato attivato.

Proseguendo la lettura del programma, troviamo la procedura **GRAFICA** che realizza il disegno dei rettangoli sul monitor; per far ciò, nei casi in cui era possibile, vengono utilizzati gli **ARRAY** precedentemente definiti. In ultimo, la procedura **TESTO** provvede alla intestazione di ogni piccolo rettangolo, permettendo così di capire a cosa serve ognuno di esso e che comando viene effettuato se viene premuto il pulsante in quel punto. Con questa **PROCEDURA** si conclude a lunga serie di sottoprogrammi; di seguito troviamo il programma principale vero e proprio, che ha il compito di richiamare le procedure principali, attivando così tutto il lavoro.

L'ultima riga del **MAIN**, permette l'abbandono del programma (e di conseguenza la conclusione del lavoro), ciò avviene qualora il tasto del mouse venga premuto nel cerchietto corrispondente al comando **QUIT**.

COMUNICAZIONE SERIALE

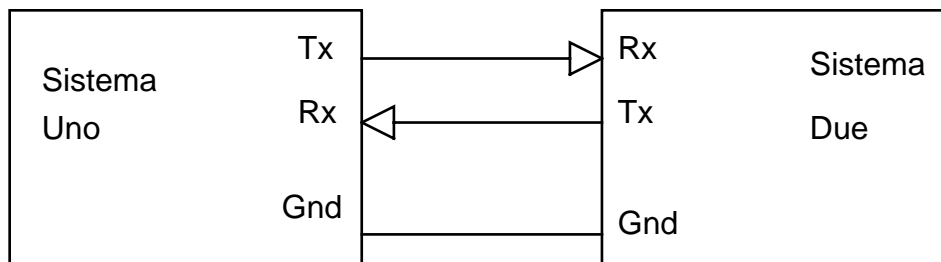
Il canale seriale **RS-232** e' il mezzo attraverso cui un dispositivo a microprocessore puo' connettersi, in modo standard, con altri dispositivi a microprocessore, personal computer, PLC, modem telefonici ed in generale con qualsiasi sistema che sia dotato di questa capacita' di comunicazione.

Un collegamento seriale **RS-232** richiede tipicamente tre fili:

- 1) La linea di **Tx** (trasmissione);
- 2) La linea di **Rx** (ricezione);
- 3) La massa comune (Gnd);

che consentono a due sistemi di scambiare informazioni.

Schematicamente:



Come si puo' vedere, la linea **Tx** di ogni sistema e' connessa alla linea **Rx** dell'altro mentre le masse sono unite.

Il principio di funzionamento di un collegamento seriale e' basato sulla scomposizione dei dati da trasmettere (byte, parole binarie) in bit inviati **SERIALMENTE** (in sequenza) **NEL TEMPO**, lungo la linea di comunicazione. Il principale vantaggio della comunicazione seriale e' il ridottissimo numero di linee necessarie. Si tratta di un vantaggio molto importante: si pensi al risparmio di cavi (e di denaro) in un collegamento tra sistemi molto distanti. Lo svantaggio e' il tempo richiesto, infatti i bit

vengono trasmessi sulla stessa linea fisica uno dopo l'altro e ciò richiede maggior tempo.

L'alternativa alla comunicazione seriale è la comunicazione **PARALLELA**. In questo tipo di collegamento sono utilizzati almeno **DIECI** fili oltre alla massa comune (in realtà le linee sono ancora di più, perché sono previsti anche segnali addizionali di controllo, sincronismo e segnalazione errore). L'esempio più comune di comunicazione parallela si trova nel collegamento PC-stampante: in tale tipo di collegamento, otto delle linee richieste trasmettono il byte di informazione (linee "**DATA**"), una ("**STROBE**") convalida l'informazione presente sulle linee "**DATA**" e l'ultima è usata dalla stampante per segnalare al computer la sua condizione di "**READY**" (condizione che abilita l'invio dei dati da parte del computer: quando la stampante non è "**READY**" il PC deve attendere). La comunicazione parallela è spesso **UNIDIREZIONALE** e questo rappresenta la causa principale per cui non è stato utilizzato da noi, poiché il nostro scopo era di creare una comunicazione bidirezionale tra PC e microprocessore.

La comunicazione parallela è naturalmente molto più veloce della comunicazione seriale ma anche più costosa; si comprende quindi come il collegamento parallelo sia utilizzato per brevi distanze, minori di 2 metri, oppure nei casi in cui l'elemento determinante sia la velocità.

Gli elementi fondamentali di una comunicazione **RS-232** sono:

- 1) **II LIVELLO FISICO** dei segnali, vale a dire la tensione o la corrente che caratterizza tali segnali. Un collegamento **RS-232** può avvenire con segnali **TTL** (ma solo su brevi distanze), con segnali $\pm 12V$ (si parla in questo caso di **RS-232C**), con segnali in corrente ("**current loop**" a 10, 20, 30 mA). Nella definizione del livello fisico è inclusa la specifica relativa al **LIVELLO LOGICO**: tipicamente un segnale **ZERO** rappresenta un bit ad **UNO**, un segnale ad **UNO** un bit a livello **ZERO**.
- 2) Il secondo parametro fondamentale è la **VELOCITÀ** di trasmissione, vale a dire la velocità con cui sono emessi i bit di informazione. La velocità di trasmissione è espressa dal **BAUD RATE** o **NUMERO DI BIT AL SECONDO**. Il baud viene definito come variazione del segnale all'interno di un canale di comunicazione (in pratica il passaggio da livello logico **ZERO** ad **UNO** oppure da **UNO** a **ZERO**).

La baud rate esprime quindi il massimo numero di variazioni che un segnale puo' avere in un secondo relativamente ad un certo canale di comunicazione. Si noti che la baud rate **NON** esprime un numero di bit al secondo anche se in pratica si tende ad assimilare il baud al bit; affermare che un certa comunicazione avviene a 1200 baud equivale quindi (praticamente) ad affermare che vengono trasferiti 1200 bit al secondo. Il protocollo **RS-232** fissa alcune velocita' standard per le trasmissioni; le piu' comuni sono:

300, 600, 1200, 2400, 4800, 9600, 19200 baud

Non esistono dispositivi standard che dialoghino, per esempio, a 1000 o 5000 baud (anche se sono previste alcune velocita' intermedie allo standard quali 1800, 3600, 7200 baud); per quanto concerne il nostro caso possiamo parlare di **9600** baud.

Inoltre è necessario specificare che la trasmissione dei dati avviene secondo un preciso ordine, dal bit meno significativo (MSB) **B0**, al bit più importante (LSB) **B7**.

3) Il terzo parametro fondamentale dell'**RS-232** e' il **FORMATO DI TRASMISSIONE** vale a dire il **NUMERO DI BIT** della parola da trasmettere (per ogni parola), il **BIT DI START** (sempre e solo 1), il numero di **BIT DI STOP**, l'eventuale **BIT DI PARITA'** o di **DISPARITA'**. Il bit di **START** e' il segnale che origina **SEMPRE** la trasmissione di una parola, il bit di **STOP** e' il segnale che la termina **SEMPRE**. Si possono avere uno oppure due bit di **STOP**. Senza bit di **START** e di **STOP** sarebbe impossibile per il ricevente capire quale sia l'inizio di una trasmissione, quale sia il bit zero di un byte, quale ne sia il bit sette. Se si stabilisce che il bit di **START** sia **ZERO** ed il bit di **STOP UNO**, l'informazione sara' certamente delimitata da uno **ZERO** iniziale e da un **UNO** finale. Il **NUMERO DI BIT DI DATO** specifica la dimensione della parola da trasmettere: tipicamente si trasmettono parole di 7 od 8 bit .

Il bit di **PARITA'** o **DISPARITA'** e' opzionale: si tratta di un elemento introdotto per aumentare l'affidabilita' della comunicazione, ciò avviene verificando la condizione di parita', se il numero di bit ad **UNO** nel dato ricevuto e' **PARI** la trasmissione e' avvenuta correttamente, se tale numero e' **DISPARI** allora si e' in presenza di errore.

E' da notare che l'aggiunta del bit di parita' **ALLUNGA** la parola da trasmettere e quindi **AUMENTA** il tempo richiesto per trasmettere i

dati.

Per aumentare ulteriormente l'affidabilità di un collegamento seriale, si possono impiegare (anche contemporaneamente), più tecniche di controllo: parità particolari, check-sum (cioè la somma di verifica, quello da noi utilizzato), oppure integrazioni matematiche all'informazione ecc.

- 4) Il quarto parametro fondamentale di una comunicazione **RS-232** è il **PROTOCOLLO DI COMUNICAZIONE**. Il protocollo di comunicazione specifica la struttura dei dati che devono essere scambiati e l'organizzazione dell'intera trasmissione.

Ad esempio si può stabilire che l'unità **"MASTER"** (principale), dialoghi con lo **"SLAVE"** (unità secondaria) attraverso particolari **STRINGHE** (sequenze) di caratteri e che i dati da trasmettere/ricevere siano organizzati in **PACCHETTI**. Le stringhe che comandano la comunicazione ed i pacchetti dati trasferiti contengono tipicamente, oltre a byte di convalida delle informazioni (checksum, riscontri calcolati con particolari algoritmi matematici ecc.), anche speciali caratteri di controllo quali : **"STX"** ("*Start of transmission*"), **"ETX"** ("*End of transmission*"), **"XON"** ("*Trasmissione abilitata*"), **"XOFF"** ("*Trasmissione disabilitata*") e parecchi altri.

Un protocollo di comunicazione deve anche gestire l'eventualità che lo slave sia impossibilitato a rispondere, o che una richiesta sia stata ricevuta erroneamente: per le diverse eventualità devono essere indicati dei procedimenti (esempio: rappresentazione richiesta "n" volte, attesa risposta per un tempo massimo definito ecc.).

Nel nostro caso specifico, il protocollo è costituito da una serie di comandi (stringhe) inviati dal computer al microprocessore e si riferiscono ai movimenti da far compiere ai motori passo-passo; nelle stringhe vengono definite una serie di parametri, tra cui quelli di inizio e di fine testo (**STX,ETX**), lo spostamento dei motori (sull'asse **X** o **Y**) e il loro verso di rotazione (sinistra, destra, alto o basso in base ai caratteri +/-). Inoltre viene definito lo stato del magnete, se acceso (**1**), se spento (**0**) e la somma di controllo (cks) cioè la somma dei valori esadecimali attribuiti dal codice **ASCII** alle singole istruzioni presenti nella stringa.

Di seguito riportiamo gli esempi di stringhe che utilizzeremo nella stesura del programma, divise a seconda della loro funzione:

SPOSTAMENTI

STX X +20 99 ETX - Sposta sull'asse X di venti unit...
verso destra.

STX X -15 96 ETX - Sposta sull' asse X di quindici unit...
verso sinistra.

STX Y +10 90 ETX - Sposta sull' asse Y di dieci unit...
verso l'alto.

STX Y -12 94 ETX - Sposta sull' asse Y di dodici unit...
verso il basso.

STATO DEL MAGNETE

STX M 1 50 ETX - Magnete in stato on.

STX M 0 4F ETX - Magnete in stato off.

PUNTO DI RIPOSO

STX O 51 ETX - Punto corrispondente all'origine degli
assi.

PUNTO DI RILASCIO DELL' OGGETTO

STX C 45 ETX - Trasporta e rilascia l'oggetto nel punto
prestabilito (contenitore).

Dopo aver ricevuto le stringhe dal computer, il microprocessore calcola la somma di controllo (**CKS**), se il risultato è uguale a quello del PC, manda indietro un segnale di "OK" (definito **ACK**) e comunica alla scheda elettronica successiva quali operazioni devono compiere i motori, in caso contrario, se il controllo non corrisponde, manda al PC un segnale di "messaggio errato" (definito **NACK**); a questo punto il calcolatore ripresenta le istruzioni per tre volte, e quasi certamente una delle tre riceverà il messaggio di ok, poichè la possibilità che si ripresenti per ben tre volte la stessa causa di errore durante la trasmissione è veramente limitata.

DESCRIZIONE SCHEDA MICROPROCESSORE

La scheda microprocessore, é una scheda appositamente progettata per far funzionare il microcomputer; é normalmente disponibile in commercio già comprensive di memorie, timer e tutti i dispositivi necessari al regolare funzionamento del microelaboratore.

Il microelaboratore é un dispositivo elettronico che può essere utilizzato per le più svariate applicazioni, infatti esso può essere “istruito” tramite una apposito programma a svolgere la determinata operazione che gli si vuole far compiere.

Come tutti i sistemi in logica programmata, anche la nostra scheda comprende le seguenti parti:

- **UNITA' CENTRALE (CPU: Central Process Unit)**, che svolge il compito di elaborazione dei dati e delle istruzioni, controllando lo sviluppo del lavoro nel nostro caso, così come nella maggior parte di essi, la **CPU** corrisponde al microprocessore.
- **MEMORIE**, si distinguono in permanenti o volatili; nelle prime (di sola lettura) vengono immagazzinate le istruzioni che consentono al sistema di eseguire le elaborazioni richieste sui dati.
Nelle memorie volatili, oltre ai dati da elaborare, vengono immagazzinati anche i risultati intermedi delle elaborazioni, ovvero quei risultati che sono destinati ad essere ulteriormente utilizzati dal sistema per altre elaborazioni.
- **UNITA' D'INGRESSO**, sono sistemi elettronici che rendono possibile il colloquio tra il mondo esterno ed il sistema; inoltre adegua i dati alle norme imposte dal collegamento (nel nostro caso é di tipo Seriale) tra il trasduttore e il sistema tramite i protocolli.
- **BUS DATI (data bus)**, é l'elemento di collegamento attraverso cui transitano i dati e le istruzioni; in particolare sul bus dati le parole transitano dalla **CPU** alle memorie e alle unità di ingresso e uscita o viceversa.
- **BUS CONTROLLI (control bus)**, che é un insieme di conduttori su ciascuno dei quali si muove un'informazione binaria corrispondente a segnali di controllo del funzionamento e disponibilità delle varie parti del sistema.,

- **BUS INDIRIZZI** (*address bus*), é un collegamento unidirezionale che va dal microprocessore alle altre parti del sistema. Su di esso viene inviata una parola binaria che consente di individuare una locazione di memoria oppure una unità d'ingresso o uscita.

Come già specificato, per poter utilizzare il microprocessore secondo i nostri scopi, abbiamo dovuto elaborare un programma in **ASSEMBLER** che gestisce tutti gli spostamenti, il loro verso, la lunghezza e la velocità di esso.

Infatti l'**ASSEMBLER** viene utilizzato quando si vuole che l'esecuzione di un programma sia molto rapida (essendo più veloce di programmi come ad esempio il **BASIC**); oppure quando si vuole che il programma occupi la minor quantità di memoria (poiché richiede meno memoria di quella richiesta ad esempio da un programma **BASIC** e suo relativo interprete).

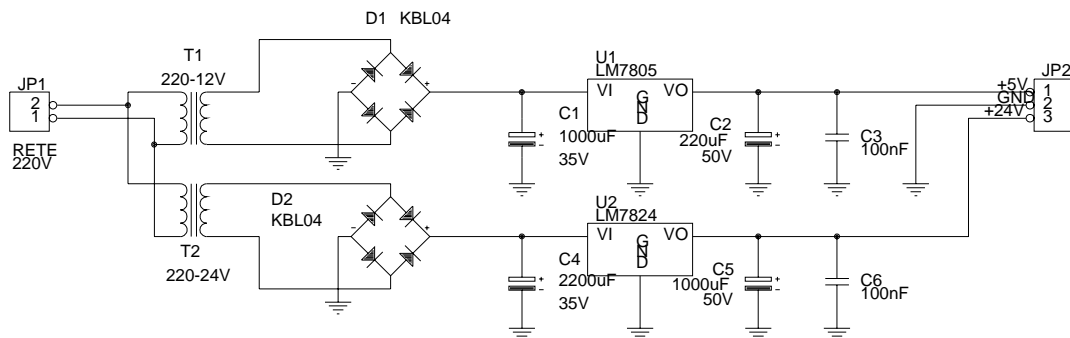
E' comunque da ricordare che l'**ASSEMBLER** è un linguaggio di programmazione a basso livello in quanto genera direttamente il codice macchina esadecimale; inoltre non ne esiste un unico tipo, ma varia a seconda del dispositivo, pur mantenendo uguali i principi di base.

La principale differenza rispetto ai linguaggi ad alto livello, è che non è trasportabile su una **CPU** di tipo diverso, cosa invece possibile per quanto riguarda linguaggi quali **BASIC, PASCAL, C**, etc.

DESCRIZIONE SCHEDA DI ALIMENTAZIONE

Lo scopo della scheda di alimentazione è quello di poter fornire i giusti valori di caduta di tensione alle schede elettroniche dei motori passo-passo pur alimentandola con la normale tensione di rete.

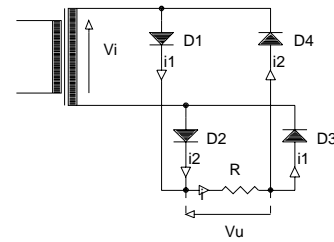
Lo schema elettrico di tale circuito è il seguente:



Come già specificato, la tensione di alimentazione del circuito è la tensione di rete 220 V, 50 HZ, di tipo sinusoidale, che si presenta agli ingressi dei due trasformatori.

La forma d' onda prelevata in uscita presenta le stesse caratteristiche di quella fornita all' ingresso ma risulta attenuata nella sua ampiezza; infatti abbiamo rispettivamente 12 V per il trasformatore **T1** e 24 V per **T2**.

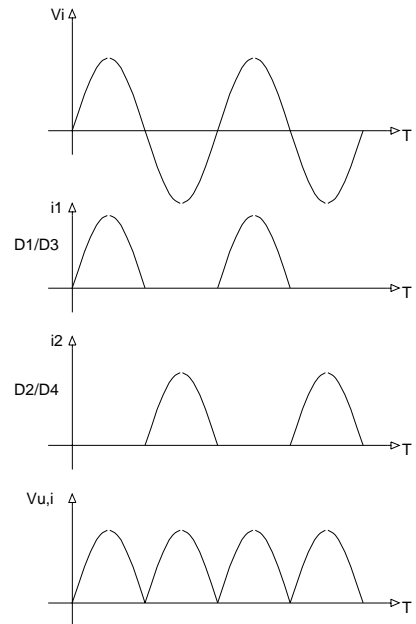
Successivamente, l'onda si presenta agli ingressi di due rispettivi raddrizzatori a ponte di Graetz, realizzati tramite una configurazione di 4 diodi, come quella visibile a fianco.



Durante la semionda positiva della tensione d'ingresso, il terminale superiore del trasformatore ha polarità positiva rispetto al terminale inferiore; i diodi **D1** e **D3** sono polarizzati direttamente e i diodi **D2** e **D4** sono interdetti. Le correnti **i1** e **i** hanno il verso segnato in figura, mentre **i2** è uguale a 0.

Durante la semionda negativa, il terminale inferiore del trasformatore ha polarità positiva rispetto al terminale superiore; i diodi **D1** e **D3** sono aperti. Le correnti **i2** e **i** hanno il verso segnato in figura mentre **i1** è uguale a 0.

In definitiva possiamo dire che la corrente nel carico **R** ha sempre lo stesso verso, ottenendo così ai capi di esso una forma d'onda pulsante.



Di seguito troviamo una serie di componenti che hanno il preciso scopo di rendere il segnale il più stabile possibile; a tale scopo utilizziamo un condensatore che riduce fortemente il ripple, rendendo l'onda più lineare.

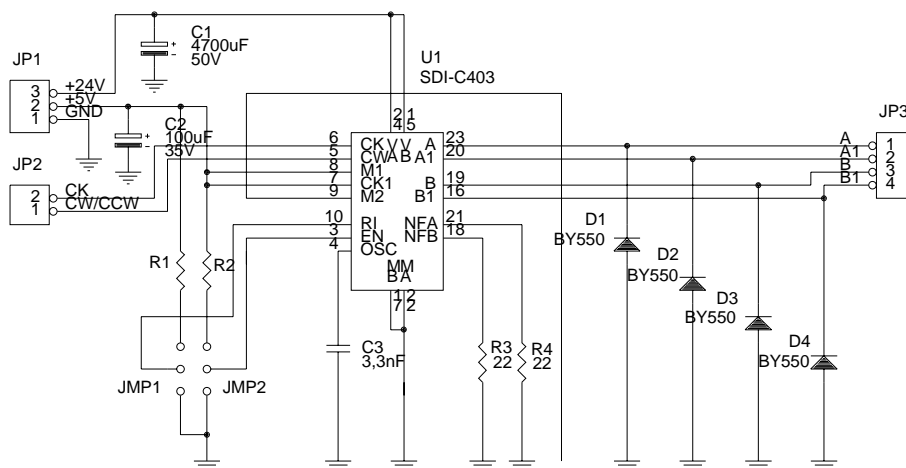
Tale segnale si presenta allo stabilizzatore di tensione integrato che la rende perfettamente continua, fissandola ad un valore prestabilito; perchè ciò sia possibile, è necessario che la tensione presente sull'ingresso sia maggiore di quella che si vuole ottenere sull'uscita, facendo attenzione a non superare i range di tensione stabiliti dal costruttore.

I condensatori posti sull'uscita dello stabilizzatore, si comportano come dei filtri, mantenendo il segnale più stabile nel tempo.

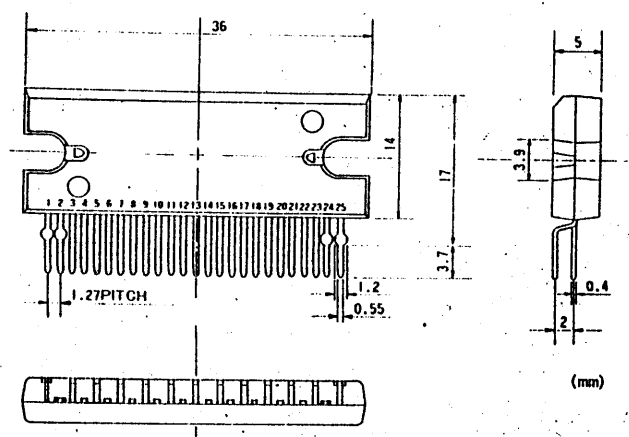
Come si può notare dallo schema elettrico, poichè le schede di pilotaggio dei motori passo-passo necessitano di essere alimentate a 5 e 24 V, sulla stessa scheda ritroviamo due circuiti.

DESCRIZIONE SCHEDA COMANDO MOTORI

Poichè i motori passo-passo necessitano di uno speciale sistema di pilotaggio, è stato necessario progettare uno speciale circuito di comando; il cui schema elettrico è il seguente:



Come si può notare, il cuore di questo sistema è il circuito integrato **SDI-403** della ditta giapponese **NMB**; come quello di figura:



Questo componente, viene utilizzato per realizzare i singoli passi da far compiere al motore, inoltre ha la capacità di far compiere 1/2, 1/4 ed arriva addirittura ad 1/8 di passo, con la conseguente riduzione delle vibrazioni ed aumento dell' efficienza.

Altre caratteristiche sono la tecnologia con cui è stato costruito, di tipo **BI-CMOS** che permette un' alta tensione di uscita, vicino ai 40 V ed una corrente media intorno a 1,5 A, fino ad arrivare a 2,5 A di picco.

Esso possiede un contenitore **CPP** a 25 piedini, con ingressi di tipo **CMOS** con resistenze di **PULL DOWN** (ancoraggio a gnd) tipicamente di 40 K Ω .

La sua alimentazione, come spiegato nella descrizione della scheda precedente è di tipo duale, per la parte integrata sono usati 5V (pin 13), mentre per la parte di pilotaggio dei motori è di 24V (pin 15,24).

Altri particolari pin che abbiamo utilizzato sono:

il pin 3 di **ENABLE**, che serve per abilitare o disabilitare le uscite del componente,

il pin 5 di **CW/CCW**, utilizzato per stabilire il verso di rotazione del motore,

i pin 6 e 7 ai quali giungono i clock di ingresso,

i pin 8 e 9 che, a seconda della loro combinazione determinano il passo eseguito dal motore, nel nostro caso 1/2 di passo.

Infine, i pin 23, 20, 19, 16 costituiscono le uscite cui sono collegati i motori.

E' ovvio che l' integrato, per il suo corretto funzionamento, abbisogna di determinati collegamenti e componenti esterni.

I condensatori **C1** e **C2** hanno la funzione di filtrare e stabilizzare al meglio la tensione di ingresso, senza subire sbalzi, le resistenze **R1** e **R2** proteggono i pin 3 e 10, rispettivamente **ENABLE** e tensione di riferimento (**REF IN**).

Per quanto riguarda i diodi **D1**, **D2**, **D3**, **D4**, hanno il compito di provvedere alla giusta polarizzazione delle bobine e permettere quindi la realizzazione del passo. Infatti gli avvolgimenti vanno attivati alternativamente, potendo realizzare quindi lo spostamento di mezzo passo tipico dei motori **HALF STEP MODE**.

Inizialmente, viene alimentata la sola fase **AB** e il rotore si dispone come indicato in fig.A.

Successivamente si attivano due fasi in modo da creare le polarità indicate in fig.B; il rotore si dispone a metà angolo fra gli avvolgimenti, ruotando di 45°.

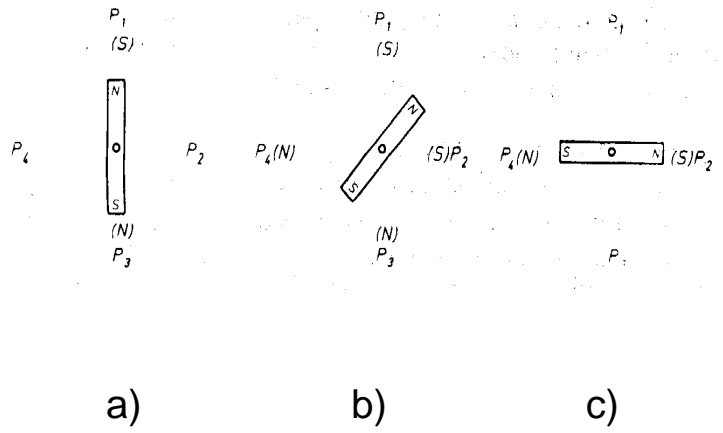
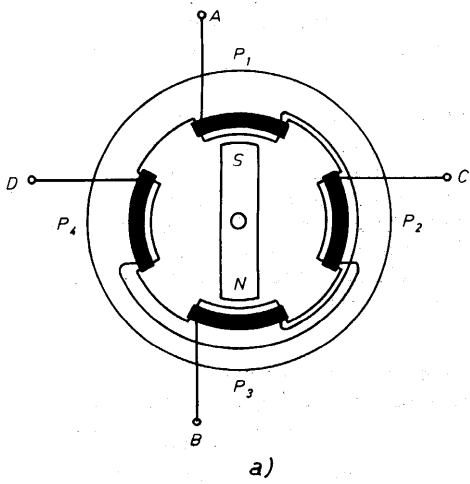
Dopodichè si ritorna al primo tipo di funzionamento, con attivazione della fase **DC** con le polarità indicate in fig.C; si ha una rotazione del rotore, ancora di 45°.

Attraverso successivi cambiamenti di modo, si ha la rotazione del rotore, con passo di 45°, quindi dimezzato rispetto agli altri due modi di funzionamento.

Anche in questo caso, per cambiare il verso di rotazione è sufficiente compiere un'inversione della sequenza dei comandi.

Questo modo di funzionamento ha il vantaggio di permettere un passo minore, ha però l' inconveniente che la coppia che sollecita il motore

non è uniforme, essendo maggiore negli istanti in cui il funzionamento è di tipo a due fasi.



USO DEL SOFTWARE

La schermata del lavoro che appare sul monitor, realizzate tramite il programma in **PASCAL**, é quella riportata nella pagina successiva.

Nella finestra più grande é visibile l'immagine che la telecamera ha inviato al computer, nelle altre sono definiti tutti i comandi possibili che possono essere effettuati.

Come già detto, il sistema permette di realizzare un automatismo per lo spostamento degli oggetti; tale scopo, può essere raggiunto secondo 3 diversi metodi di funzionamento: **AUTOMATICO**, **CONTROLLATO** o **MANUALE**.

Il modo **AUTOMATICO** si attiva premendo il pulsante del mouse nel cerchietto vicino alla scritta, a questo punto l'oggetto viene riconosciuto, ed automaticamente prelevato e portato nel contenitore posto all'angolo del piano.

Il modo di funzionamento **CONTROLLATO**, serve per effettuare gli spostamenti dell'oggetto controllandoli appunto tramite il mouse.

Questa volta però, il computer non riconosce automaticamente la posizione dell'oggetto, ma bisognerà fornirgliela premendo il pulsante del mouse sopra di esso; dopodiché, "cliccando" nel punto desiderato interno all'immagine, gli viene definita la posizione in cui deve essere portato.

Bisogna comunque specificare che, se non viene attivato il magnete premendo sull'apposita casella, il prelievo non verrà effettuato; i comandi **CASSETTO** ed **ORIGINE**, permettono rispettivamente di portare l'oggetto nel contenitore, o nel punto del piano corrispondente alle coordinate $(0,0)$.

Se invece vogliamo effettuare **MANUALMENTE** la totalità delle operazioni, basta portarsi nell'ultima parte dello schermo, e premere il pulsante nella finestra desiderata.

I comandi **SPOSTAMENTO X** e **SPOSTAMENTO Y**, permettendo di effettuare le rispettive operazioni solo se viene inserita la quantità dello spostamento, i comandi **GO** e **STOP** permettendo di attivare o disattivare il sistema, **CASSETTO** ed **ORIGINE** assolvono agli stessi compiti che avevano nel funzionamento **CONTROLLATO**, mentre **MAGNETE ON** e **MAGNETE OFF** attivano e disattivano il dispositivo di prelievo.

E' da notare che nella parte più alta dello schermo compaiono delle scritte a seconda dei tasti o del modo di funzionamento che abbiamo attivato.

In alto a sinistra viene indicato lo spostamento che l'operatore introduce tramite la tastiera, dopo aver attivato i comandi **SPOSTAMENTO X** e **SPOSTAMENTO Y** facenti parte del modo di funzionamento **MANUALE**.

Al centro vengono invece indicate le coordinate (X,Y) del punto in cui si trova l'oggetto al momento del prelievo durante i modi di funzionamento **AUTOMATICO** e **CONTROLLATO**.

Infine, in alto a destra vengono visualizzati dei commenti che spiegano sinteticamente quale operazione il sistema si accinge a compiere a seconda del comando attivato.

SOFTWARE

Program Tesi;

Uses

Graph, Mouse;

Const

```

Stx      =      #2      ;
Etx      =      #3      ;
Spazio   =      #32     ;
Rsdata   : Integer = $3F8 ;{ Registro Dati Porta Seriale           $3F8 Com1 $2F8 Com2}
Rshlatch: Integer = $3F9 ;{ Registro Seriale Latch Parte Alta Count $3F9 Com1 $2F9 Com2}
Rsreg    : Integer = $3FB ;{ Registro Indirizzam. Interno Porta Seriale   $3FB Com1 $2FB Com2}
Rsmode   : Integer = $3FC ;{ Registro Modo Funzionam. Porta Seriale   $3FC Com1 $2FC Com2}
Rsstatus : Integer = $3FD ;{ Registro Di Stato Porta Seriale     $3FD Com1 $2FD Com2}
Center_X : Array[1..14] Of Integer = (414,416,           {Quota X Del Centro Dei Cerchi}
                                     25,183,
                                     341,499,
                                     25,25,
                                     183,183,
                                     341,341,
                                     499,499);
Center_Y : Array[1..14] Of Integer = (77,360,           {Quota Y Del Centro Dei Cerchi}
                                     389,389,
                                     389,389,
                                     423,455,
                                     423,455,
                                     423,455,
                                     423,455);
Rectxmi  : Array[1..15] Of Integer = (400,400,           {Quota X Minima Dei Rettangoli}
                                     404,12,
                                     170,328,
                                     486,12,
                                     12,170,
                                     170,328,
                                     328,486,
                                     486);
Rectymi  : Array[1..15] Of Integer = (90,67,            {Quota Y Minima Dei Rettangoli}
                                     350,379,
                                     379,379,
                                     379,413,
                                     445,413,
                                     445,413,
                                     445,413,
                                     445);
Rectxma  : Array[1..15] Of Integer = (635,635,           {Quota X Massima Dei Rettangoli}
                                     629,152,
                                     310,468,
                                     626,152,
                                     152,310,
                                     310,468,
                                     468,626,
                                     626);
Rectyma  : Array[1..15] Of Integer = (345,87,           {Quota Y Massima Dei Rettangoli}
                                     369,399,
                                     399,399,
                                     399,433,
                                     465,433,
                                     465,433,
                                     465,433,
                                     465);

```

```

Var   Graphdriver      :      Integer ;
      Graphmode        :      Integer ;
      Errorcode        :      Integer ;
      G,F,S,Str_Asc,Str_Asc1 :      String ;
      X,Y,Z,I,J,K,Xm,Ym,
      Quotax,Quotay,
      Color,Button     :      Integer ;
      Tasto            : Array[1..14] Of Byte ;
      B                :      Byte ;
      Attendi          :      Longint ;

```

```

(*****
*****FUNZIONI*****
)

```

```

Function Hex_Asc(Valore:Integer):String ;           {Traduce Da Esadecimale Ad ASCII
                                                    Il Valoredello Spostamento
                                                    E Checksum}

```

```

Var
  Nibble :      Byte ;
  I       :      Integer ;
  Str_Asc :      String ;
Begin
  For I := 1 To 4 Do
  Begin
    Nibble := Valore And $F ;
    If Nibble > 9 Then
    Str_Asc[I] := Chr (Nibble+55)
    Else Str_Asc[I] := Chr (Nibble+48) ;
    Valore := Valore Div 16 ;
  End ;
  Str_Asc[5] := Chr(0) ;
  Hex_Asc := Str_Asc ;
End;

```

```

(*****
)

```

```

Function Checksum(Var S :String ): Integer;           {Calcola Il Chks Della Stringa}

```

```

Var
  I :      Integer ;
  D :      Integer ;
Begin
  D :      =      0 ;
  I :      =      1 ;
  Repeat
  D :      =      D+Ord(S[I]) ;
  I :      =      I+1 ;
  Until S[I] =      Char(Spazio) ;
  Checksum :      =      D ;
END;

```

```

(*****
)

```

```

Function Coda(Var S : String) : String ;           {Accoda Chks E Etx}

```

```

Var   App :      Integer ;
Begin
  App :      =Checksum(S) ;
  Str_Asc1 :      =Hex_Asc(App) ;
  Coda :      = S+Str_Asc1[1]+Str_Asc1[2]+Etx ;
End;

```

```

(*****
)

```

(*****PROCEDURE*****)

```
Procedure Rs_Init(V: Integer); {Inizializza Il Canale Seriale}
Begin
    Port[Rsreg] := $80 ;
    Port[Rpdata] := V ;
    Port[Rshlatch] := 0 ;
    Port[Rsreg] := 3 ;
    Port[Rsmode] := 3 ;
End;
```

(*****)

```
Procedure Rs_Output(D :CHAR) ; {Spedisce Nel Canale Seriale Un Carattere}
Begin
    While (Port[Rsstatus] And $60) <> $60 Do ;
        Port[Rpdata] :=Ord (D) ;
End;
```

(*****)

```
Procedure Rs_String_Output(S : String ) ; {Accumula I Caratteri E Li Spedisce}
Var I : Integer ;
Begin
    For I := 1 To Length(S) Do Rs_Output(S[I]) ;
End;
```

(*****)

```
Procedure Grafica; {Disegna Le Finestre Sul Video}
Begin
    Graphdriver := Detect ;
    Initgraph(Graphdriver, Graphmode, ") ;
    Errorcode := Graphresult ;
    If Errorcode <> Grok Then ;
        Begin ;
            Writeln ('Errore' , Grapherrormsg(Errorcode)) ;
            Writeln ('Programma Interrotto; Manca Il File Egavga.Bgi') ;
            Halt(1) ;
        End;
        R_Mouse(B) ;
        Setcolor(Blue) ;
        For I:=1 To 15 Do Rectangle(Rectxmi[I],Rectymi[I],Rectxma[I],Rectyma[I]) ;
        Setcolor(Yellow) ;
        For I:=1 To 14 Do Circle(Center_X[I], Center_Y[I], 7) ;
        Rectangle (3, 19, 635, 64) ;{Finestra Titolo }
        Rectangle (3, 67, 396, 372) ;{Finestra Grafica }
        Rectangle (400, 347, 635, 372) ;{Finestra Comandi Automatico }
        Rectangle (3, 374, 635, 403) ;{Finestra Comandi Controllato }
        Rectangle (3, 405, 635, 475) ;{Finestra Comandi Manuale }
        Setcolor(White) ;
        Rectangle (0, 16, Getmaxx, Getmaxy) ;
End;
```

(*****)

```
Procedure Testo; {Scrive I Testi Sul Video}
Begin
    Settextjustify (Centertext, Centertext) ;
    Settextstyle (Defaultfont, Horizdir, 2) ;
    Outtextxy (Getmaxx Div 2,33, 'I.P.S.I.A."Moretto" Maturità 95/96') ;
    Settextstyle (Defaultfont ,Horizdir ,1) ;
```

```

Outtextxy (Getmaxx Div 2 ,47, 'Automatismo Computerizzato E Monitorizzato' );
Outtextxy (Getmaxx Div 2 ,57, 'Per Il Trasporto Di Oggetti' );
Setcolor(7) ;

Outtextxy (90, 389, 'Cassetto' );

Outtextxy (248, 389, 'Origine' );

Outtextxy (401, 389, 'Magnete On' );

Outtextxy (564, 389, 'Magnete Off' );

Outtextxy (520, 77, 'Quit' );

Outtextxy (520, 360, 'Automatico' );

Outtextxy (90, 424, 'Spostamento X' );

Outtextxy (90, 456, 'Spostamento Y' );

Outtextxy (248, 424, 'Go' );

Outtextxy (248, 456, 'Errore' );

Outtextxy (401, 424, 'Cassetto' );

Outtextxy (401, 456, 'Origine' );

Outtextxy (564, 424, 'Magnete On' );

Outtextxy (564, 456, 'Magnete Off' );

```

End;

(*****)

```

Procedure Show_Imm; {Traduce Dal Formato BMP A PCX
                    L'immagine Della Telecamera}

Var
  Fromf,To      F      :      File      ;
  Numread,      Px      :      Word      ;
  Y_C, X_C, I, I3    :      Integer     ;
  Buf           :Array[1..1200] Of      Char      ;

Const
  DELTA_X      =      4      ;
  DELTA_Y      =      67     ;

Begin
  Assign(Fromf, '1.Bmp') ;
  Reset(Fromf, 1) ;
  Seek(Fromf,54) ;
  Y_C := 304 ;
  Repeat
    Blockread(Fromf, Buf, 1184, Numread) ;
    X_C := 0 ;
    For I := 1 To 392 Do
      Begin
        I3 := I * 3 ;
        Px := Ord(Buf[I3]) + Ord(Buf[I3+1]) + Ord(Buf[I3+2]) ;
        Px := Px Div(3) ;
        If Px > 127 Then
          Px := White ;
        Else Px := BLACK ;
      End ;
    End ;
End ;

```

```

                Putpixel(X_C+Delta_X, Y_C+Delta_Y, Px) ;
                X_C := X_C + 1 ;
            End;
            Y_C := Y_C - 1 ;
            Until Y_C < 1 ;
        Close(Fromf) ;
    End;

```

(*****)

```

Procedure M_On; {Spedisce La Stringa M_On}
Begin
    S := Stx+'M'+1'+Char(Spazio) ;
    S := Coda(S) ;
    Rs_String_Output(S) ;
End;

```

(*****)

```

Procedure M_Off; {Spedisce La Stringa M_Off}
Begin
    S := Stx+'M'+0'+Char(Spazio) ;
    S := Coda(S) ;
    Rs_String_Output(S) ;
End;

```

(*****)

```

Procedure Spostamento_X(Var Quotax : Integer); {Spedisce La Stringa
                                                Per Lo Spostamento Su X}
Var Segno : Char ;
Begin
    Str_Asc := Hex_Asc(Quotax) ;
    If Quotax >= 0 Then Segno := '+' Else Segno := '-' ;
    Quotax := Abs(Quotax) ;
    S := Stx+'X'+Segno+Str_Asc[1]+Str_Asc[2]+Str_Asc[3]+Str_Asc[4]+Char(Spazio);
    F := Coda(S) ;
    Rs_String_Output(F) ;
End;

```

(*****)

```

Procedure Spostamento_Y(Var Quotay : Integer); {Spedisce La Stringa
                                                Per Lo Spostamento Su Y}
Var Segno : Char ;
Begin
    Str_Asc := Hex_Asc(Quotay) ;
    If Quotay >= 0 Then Segno := '+' Else Segno := '-' ;
    Quotay := Abs(Quotay) ;
    S := Stx+'Y'+Segno+Str_Asc[1]+Str_Asc[2]+Str_Asc[3]+Str_Asc[4]+Char(Spazio);
    G := Coda(S) ;
    Rs_String_Output(G) ;
End;

```

(*****)

```

Procedure Origine; {Spedisce La Stringa
                  Magnete Nel Punto 0,0 }
Begin
    S := Stx+'O'+Char(Spazio) ;
    S := Coda(S) ;
    Rs_String_Output(S) ;

```

End;

(*****)

Procedure Contenitore; {Spedisce La Stringa
Per Portare L'oggetto
Nel Contenitore}

Begin
S := Stx+'C'+Char(Spazio) ;
S := Coda(S) ;
Rs_String_Output(S) ;
End;

(*****)

Procedure Errore; {Spedisce La Stringa
Per Introdurre
Un Errore Nel Up}

Begin
S := Stx+'E'+Char(Spazio) ;
S := Coda(S) ;
Rs_String_Output(S) ;
End;

(*****)

Procedure Leggi_Spostamento; {Legge Per La Procedura
Automatico Il Valore
Dello Spostamento}

Begin
Y := 72 ;
Repeat
Y := Y+1 ;
X := 4 ;
Repeat
X := X+1 ;
Color := Getpixel(X,Y) ;
Until (Color=0)Or(X=395) ;
Until (Color=Black)OR(Y=372) ;
Quotax := X-4 ;
Quotay := Y-68 ;
End;

(*****)

Procedure Leggi_Mouse; {E'la Procedura Principale
Del Programma Serve
A Riconoscere Le
Clickate Del
Mouse}

Begin
For I:=1 To 14 Do
Tasto[I] := 0 ;
Repeat
G_Mouse(Button,Xm,Ym) ;
Until Button = 1 ;
For I := 1 To 14 Do
If (Abs(Xm-Center_X[I]) <> 10) And (Abs(Ym-Center_Y[I]) <> 10) Then
Begin
H_Mouse ;
Setfillstyle(1,Black) ;
Setcolor(Yellow) ;
End;

```

Fillellipse(Center_X[I],Center_Y[I],7,7) ;
S_Mouse ;
If (Abs(Xm-Center_X[I]) < 10) And (Abs(Ym-Center_Y[I]) < 10) Then
Begin
    H_Mouse ;
    Tasto[I] := 1 ;
    Setfillstyle(1,Red) ;
    Setcolor(Yellow) ;
    Fillellipse(Center_X[I],Center_Y[I],7,7) ;
    S_Mouse ;
End;
End;
Settextjustify (Centertext , Centertext ) ;
Settextstyle (Defaultfont , Horizdir , 1) ;
(**) If (Abs(Xm-Center_X[2]) < 10) And (Abs(Ym-Center_Y[2]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Eseguo La Procedura Automatica') ;
    H_Mouse ;
    Leggi_Spostamento ;
    S_Mouse ;
    Outtextxy(250,9,'X=') ;
    Str(Quotax,S) ;
    Outtextxy(270,9,S) ;
    Outtextxy(300,9,'Y=') ;
    Str(Quotay,S) ;
    Outtextxy(320,9,S) ;
    Spostamento_X(Quotax) ;
    Outtextxy(500,9,'Spostamento Asse X') ;
    Grafica ;
    Testo ;
    Show_Imm ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
    Spostamento_Y(Quotay) ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
    M_On ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;

```

```

Contentitore ;
Attendi := 50000000 ;
Repeat
    Attendi := Attendi - 1 ;
Until Attendi = 0 ;
M_Off ;
Attendi := 50000000 ;
Repeat
    Attendi := Attendi - 1 ;
Until Attendi = 0 ;
Origine ;
End;
(**) If (Abs(Xm-199) < 196) And (Abs(Ym-217) < 155) And (Button =1) Then
Begin
    Outtextxy(500,9,'Eseguo La Procedura Controllata') ;
    Outtextxy(300,9,'X=') ;
    Str(Xm,S) ;
    Outtextxy(320,9,S) ;
    Outtextxy(250,9,'Y=') ;
    Str(Ym,S) ;
    Outtextxy(270,9,S) ;
    K := 1 ;
    Xm := Xm*K ;
    Spostamento_X(Xm) ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
    Ym := Ym*K ;
    Spostamento_Y(Ym) ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
End;
If (Abs(Xm-Center_X[3]) < 10) And (Abs(Ym-Center_Y[3]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Portare Oggetto Nel Contenitore') ;
    Contentitore ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
End;
If (Abs(Xm-Center_X[4]) < 10) And (Abs(Ym-Center_Y[4]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Ritornare Alla Posizione Di Riposo') ;
    Attendi := 50000000 ;
    Repeat
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
    Origine ;
End;
If (Abs(Xm-Center_X[5]) < 10) And (Abs(Ym-Center_Y[5]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Attivare Il Magnete') ;
    M_On ;
    Attendi := 50000000 ;
    Repeat

```

```

                Attendi          :=      Attendi - 1          ;
            Until Attendi          =      0                    ;
End;
If (Abs(Xm-Center_X[6]) < 10) And (Abs(Ym-Center_Y[6]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Disattivare Il Magnete')                ;
    M_Off                                                    ;
    Attendi          :=      50000000                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1                ;
    Until Attendi          =      0                            ;
End;
(**) If (Abs(Xm-Center_X[7]) < 10) And (Abs(Ym-Center_Y[7]) < 10) And (Button =1) Then
Begin
    Outtextxy (550,9,'Spostamento X Max 393')                ;
    Readln(Quotax)                                           ;
End;
If (Abs(Xm-Center_X[8]) < 10) And (Abs(Ym-Center_Y[8]) < 10) And (Button =1) Then
Begin
    Outtextxy (550,9,'Spostamento Y Max 305')                ;
    Readln(Quotay)                                           ;
End;
If (Abs(Xm-Center_X[9]) < 10) And (Abs(Ym-Center_Y[9]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Eseguo La Procedura Manuale')            ;
    Spostamento_X(Quotax)                                    ;
    Attendi := 50000000                                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1;                ;
    Until Attendi          =      0                            ;
    Spostamento_Y(Quotay)                                    ;
    Attendi          :=      50000000                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1                ;
    Until Attendi          =      0                            ;
End;
If (Abs(Xm-Center_X[10]) < 10) And (Abs(Ym-Center_Y[10]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Blocca Tutte Le Operazioni In Atto')    ;
    Attendi          :=      50000000                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1                ;
    Until Attendi          =      0                            ;
    Stop                                                       ;
End;
If (Abs(Xm-Center_X[11]) < 10) And (Abs(Ym-Center_Y[11]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Portare Oggetto Nel Contenitore')        ;
    Contenitore                                             ;
    Attendi          :=      50000000                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1                ;
    Until Attendi          =      0                            ;
End;
If (Abs(Xm-Center_X[12]) < 10) And (Abs(Ym-Center_Y[12]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Ritornare Alla Posizione Di Riposo')    ;
    Attendi          :=      50000000                        ;
    Repeat                                                    ;
        Attendi          :=      Attendi - 1                ;
    Until Attendi          =      0                            ;
    Origine                                                  ;

```

```

End;
If (Abs(Xm-Center_X[13]) < 10) And (Abs(Ym-Center_Y[13]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Azionare Il Magnete') ;
    M_On ;
    Attendi := 50000000 ;
    Repeat ;
        Attendi := Attendi - 1 ;
    Until Attendi =0 ;
End;
If (Abs(Xm-Center_X[14]) < 10) And (Abs(Ym-Center_Y[14]) < 10) And (Button =1) Then
Begin
    Outtextxy(500,9,'Disattivare Il Magnete') ;
    M_Off ;
    Attendi := 50000000 ;
    Repeat ;
        Attendi := Attendi - 1 ;
    Until Attendi = 0 ;
End;
End;

(*****
*****MAIN*****
*****)
{Programma Principale;
Richiama Tutte Le Altre Routine}

Begin
Rs_Init(12) ;
Repeat ;
    Grafica ;
    Testo ;
    Show_Imm ;
    S_Mouse ;
    Leggi_Mouse ;
Until (Abs(Xm-Center_X[1])<10)And(Abs(Ym-Center_Y[1])<10)And(Button =1) ;
Closegraph ;
End.

(*****)

```

PRESENTAZIONE	2
ARCHITETTURA DEL SISTEMA	3
CENNI SUL COMPUTER	5
SOFTWARE DEL SISTEMA	6
COMUNICAZIONE SERIALE	9
DESCRIZIONE SCHEDA MICROPROCESSORE	14
DESCRIZIONE SCHEDA DI ALIMENTAZIONE	16
DESCRIZIONE SCHEDA COMANDO MOTORI	18
USO DEL SOFTWARE	21
SOFTWARE	23