

IMPARARE A GESTIRE LA PORTA PARALLELA

Da oggi non sarà più un problema sfruttare le potenzialità della porta parallela di un computer, perché vi spiegheremo tutti i meccanismi per usarla nel migliore dei modi

Paolo Sbrana - 1^a parte

Da quando il computer è entrato nella maggior parte delle case degli italiani, si sta cercando di sfruttarlo in ogni modo possibile, assegnandogli compiti di scrittura, segnalazione, gestione di allarmi e di periferiche, controllo di processi più o meno automatizzati ecc.

Per dialogare con questa "scatola", ci sono due vie principali denominate "porta seriale" e "porta parallela". Per la prima però, dobbiamo nel 99% dei casi abbinare almeno un microcontroller per gestire la comunicazione asincrona, con conseguente aumento della complessità circuitale.

Per la porta parallela, invece, è possi-

bile adottare un hardware veramente minimo unito ad un software ancora più semplice.

Lo scopo di questa serie di articoli è infatti far capire come sfruttare al meglio la porta parallela, anche per chi non ha molta conoscenza di linguaggi di programmazione.

Fondamentalmente, vedremo come sia possibile ottenere delle uscite digitali, eventuali uscite analogiche e ingressi digitali.

In questa prima parte, avremo modo di interfacciare un integrato generatore di DTMF (Dual Tone Multi Frequency) e successivamente dei Led o dei relè per le più svariate applicazioni.

La generazione dei toni DTMF

Abbiamo già proposto in passato dei circuiti che consentivano di gestire toni DTMF attraverso il computer, ma attraverso la porta seriale e quindi con in più almeno un chip dedicato al colloquio.

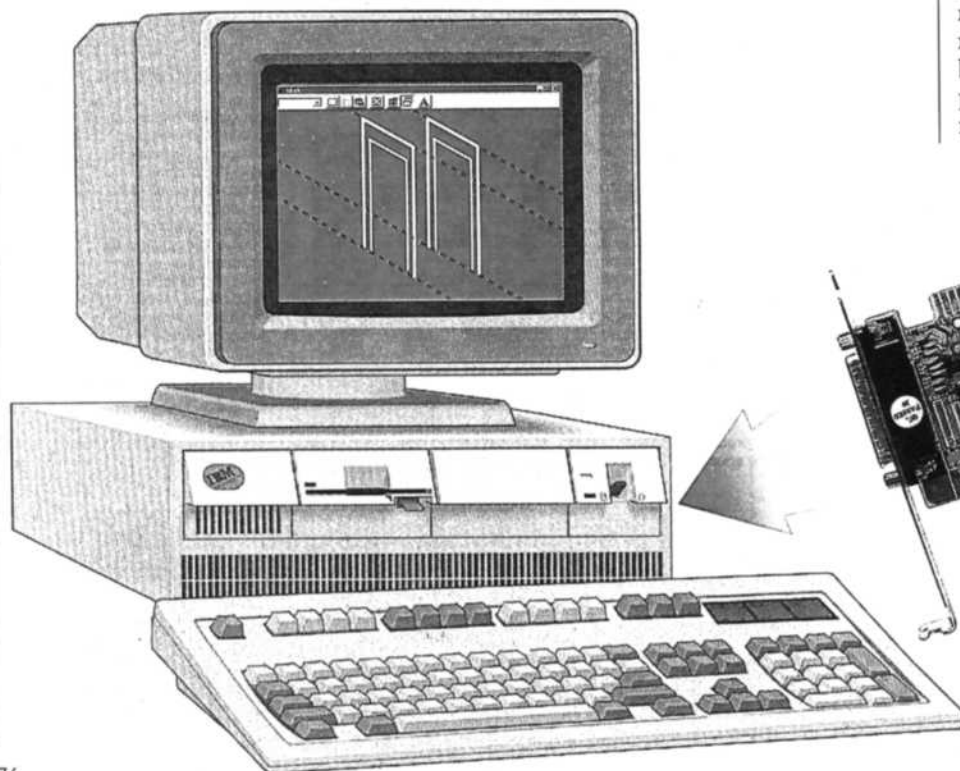
Prima di andare a vedere la nostra ultima soluzione però, dobbiamo spiegare che è possibile paragonare la porta parallela di un computer all'integrato SN74374, ovvero ad un latch ad 8 vie (per quanto riguarda la sezione di output).

Con il software, possiamo modificare il valore di questi latch quando vogliamo, con l'unica restrizione della velocità strettamente correlata alla "potenza" del computer impiegato.

A parità di caratteristiche del 74374, non è possibile modificare solo il valore di una singola uscita, ma devono essere riscritti sulla porta sempre 8 bit.

Detto questo, andiamo a vedere in Figura 1 lo schema relativo alla nostra prima applicazione della porta parallela.

I segnali dei dati e di controllo vengono prelevati direttamente dai pin relativi ai dati di uscita della parallela (D0, D1, ..., D7) ed inviati sul chip TP5088 della National. Questo integrato genera autonomamente i sedici toni standard DTMF visibili nella Tabella 1 in funzione dei livelli presenti sui suoi ingressi (sempre con riferimento alla Tabella 1). Lo schema a



**Scheda
porta
parallela**

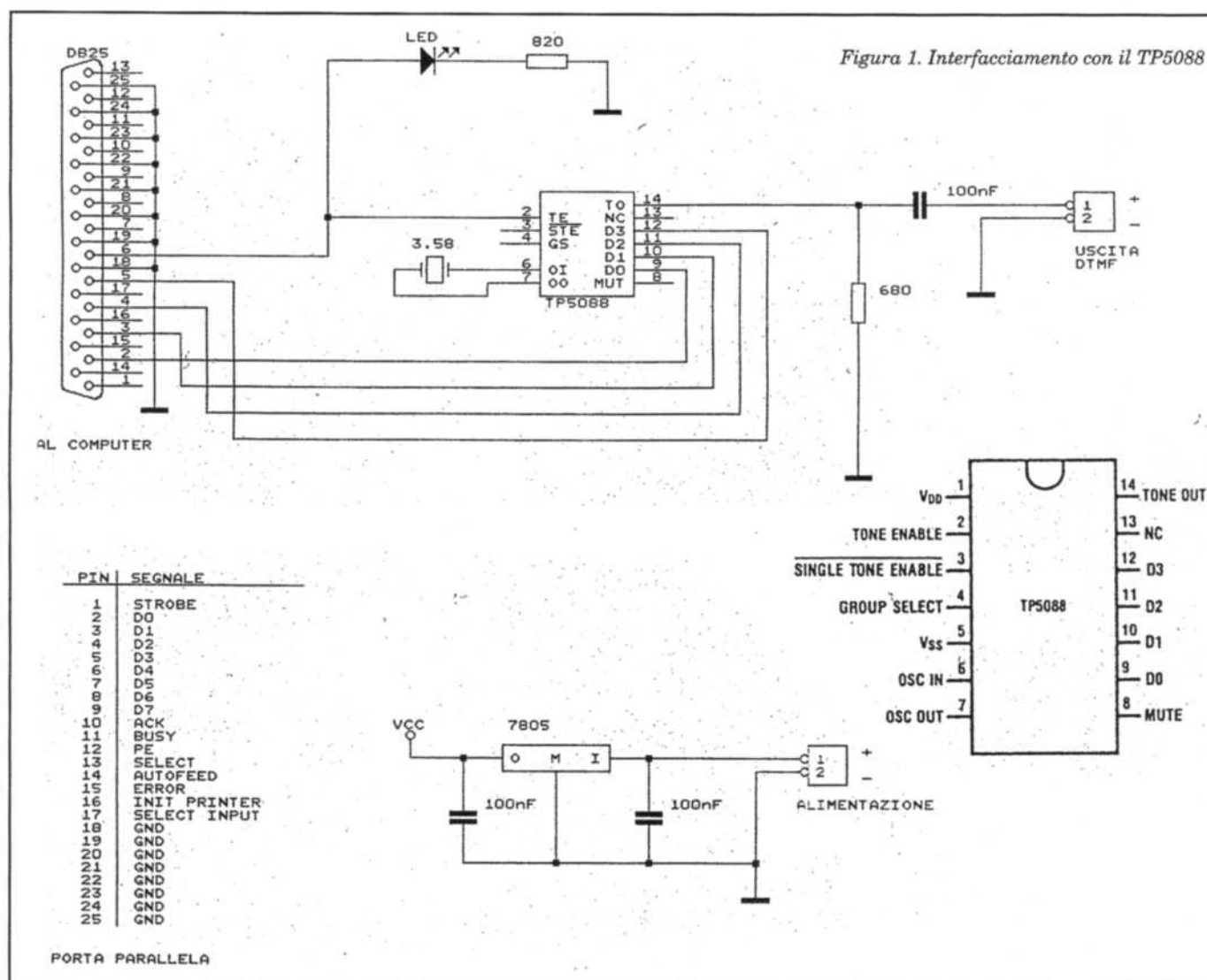


Figura 1. Interfacciamento con il TP5088

Tabella 1. Relazione tra toni DTMF e frequenze

Keyboard Equivalent	Data Inputs				TONE ENABLE	TONES OUT		MUTE
	D3	D2	D1	D0		f _L (Hz)	f _H (Hz)	
X	X	X	X	X	0	0V	0V	0V
1	0	0	0	1	/	697	1209	O/C
2	0	0	1	0	/	697	1336	O/C
3	0	0	1	1	/	697	1477	O/C
4	0	1	0	0	/	770	1209	O/C
5	0	1	0	1	/	770	1336	O/C
6	0	1	1	0	/	770	1477	O/C
7	0	1	1	1	/	852	1209	O/C
8	1	0	0	0	/	852	1336	O/C
9	1	0	0	1	/	852	1477	O/C
0	1	0	1	0	/	941	1336	O/C
*	1	0	1	1	/	941	1209	O/C
#	1	1	0	0	/	941	1477	O/C
A	1	1	0	1	/	697	1633	O/C
B	1	1	1	0	/	770	1633	O/C
C	1	1	1	1	/	852	1633	O/C
D	0	0	0	0	/	941	1633	O/C

blocchi di tale chip è visibile in Figura 1: un oscillatore interno controllato da un quarzo esterno da 3,579545 MHz, u data latch in cui memorizzare i bit d'ingresso, due gruppi identici strutturalmente ma con valori diversi che generano le frequenze DTMF impostate, un sezione per il controllo del MUTE infine uno stadio amplificatore di uscita comandato da un segnale di abilitazione esterno (lo stesso che controlla il latch dei dati d'ingresso).

Per far generare allora un tono DTMF è necessario impostargli il valore relativo al tono prescelto sugli ingressi D0, D1, D2 e D3 e poi portare ad alto livello il pin 2 (Tone Enable). Per tutto il tempo in cui questo pin viene tenuto ad alto livello, sull'uscita 14 sarà presente il tono DTMF impostato.

Vediamo, quindi, il piccolo programma denominato TP5088.C che proponiamo (PROGRAMMA 1). Come si vede è un programma scritto in linguaggio "C

PROGRAMMA 1

```

/
*#####*/
/* GESTIONE DELLA PORTA PARALLELA PER INVIO DI TONI
DTMF */
/
*#####*/
#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

#define LPT1 0x378 /* indirizzo stampante */

#define TONOOFF 0x00 /* disattiva tono DTMF */
#define TONOON 0x10 /* attiva tono DTMF */

typedef unsigned char BYTE;

int main(int argc, char *argv[])
{
    int i;

    outp(LPT1,TONOOFF); /* disabilita tono DTMF */
    for(i=0;i<=0x0f;i++) /* ciclo dal carattere D al C */
    {
        outp(LPT1,i); /* predisporre tono da inviare */
        delay(1); /* attesa 1 ms */
        outp(LPT1,TONOON); /* attiva tono */
        delay(100); /* attesa 100 ms */
        outp(LPT1,TONOOFF); /* disattiva tono */
        delay(100); /* attesa 100 ms */
    }
    return 0;
}

```

PROGRAMMA 2

```

/
*#####*/
/* GESTIONE DELLA PORTA PARALLELA PER CONTROLLO LED
*/
/
*#####*/
#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

#define LPT1 0x378 /* indirizzo stampante */

#define LEDOFF 0x00 /* spegna tutti led */

typedef unsigned char BYTE;

int main(int argc, char *argv[])
{
    int i;

    outp(LPT1,LEDOFF); /* spegna tutti i led */
    for(i=1;i<=0x80;i*=2) /* accendo dal led 1 al led 8 */
    {
        outp(LPT1,i); /* accende led */
        delay(200); /* attesa 200 ms */
    }
    outp(LPT1,LEDOFF); /* spegna tutti i led */
    return 0;
}

```

e compilato con Borland 3.0 in italiano. Tale scelta è stata dettata dal fatto che ormai in molti conoscono tale linguaggio ed in particolar modo è molto facile scrivere sulla porta parallela.

Infatti, se si vanno a contare le righe di programmazione effettiva, notiamo che sono solamente 9!

L'effetto di questo programma è far generare tutti i toni DTMF a partire dal

D(0h) per arrivare al C(fh) con durata di 100 ms ciascuno e pausa tra toni di altri 100ms. Per variare questi valori o per ottenere altri tipi di gestione, sarà sufficiente modificare il programma.

Ad esempio, dopo aver eseguito qualche prova, non vi sarà difficile inserire anche un relè e così implementare un apparato che consenta di aprire e chiudere una linea telefonica e successivamente

e 0xf. Lo stesso risultato lo avremmo ottenuto anche scrivendo "outp (LPT1,TONO ON +i)", ovvero sommando i due singoli valori.

Il controllo delle uscite

Proseguendo con i nostri esperimenti, vediamo come sia possibile comandare fino ad 8 carichi distinti collegandoci direttamente sulla porta parallela.

In Figura 3 possiamo vedere il semplicissimo circuito con cui andare a posizionare 8 Led sulla porta tramite 8 resistenze di limitazione della corrente. Ovviamente abbiamo scelto questo tipo di uscite per consentire a tutti una rapida realizzazione, ma sarà poi possibile controllare anche carichi di diverso tipo, come relè (vedi schema di Figura 4), triac, fotoaccoppiatori, ecc.

Analizziamo subito il programma LED.C scritto appositamente (PROGRAMMA 2). Anche in questo caso le righe scritte sono appena 6, per ottenere l'accensione di un Led alla volta a partire dal primo. Ovviamente, lo scopo di questo programma è quello di far capire a tutti le potenzialità della porta parallela e non di permettere la gestione di una macchina a controllo numerico, ma da qui è possibile partire anche per arrivare a soluzioni di questo tipo.

Una tra le applicazioni più frequentemente richieste, è il controllo di gruppi di luce, sia per manifestazioni ludiche

PROGRAMMA 3

```

/
*#####*/
/* COMPOSIZIONE DI UN NUMERO TELEFONICO IN PULSE
*/
/
*#####*/
#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

#define LPT1 0x378 /* indirizzo stampante */

#define LINEAON 0x01 /* chiudi linea */
#define LINEAOFF 0x00 /* aprì linea */

typedef unsigned char BYTE;

int main(int argc, char *argv[])
{
    int i,j;

    char numero[20];

    sprintf(numero,"02660251\0");
    outp(LPT1,LINEAON); /* impegna linea */
    delay(2000); /* attesa 2 secondi */
    for(i=0;i<strlen(numero);i++) /* compongo numero */
    {
        if(numero[i]==0) /* se cifra = 0 */
            numero[i]=10; /* inserisco 10 */
        for(j=0;j<numero[i];j++)
        {

```

di comporre dei numeri sia in decadico che in multifrequenza.

Merita però soffermarci sul metodo impiegato per scrivere sulla porta il carattere DTMF da inviare: l'istruzione che si incarica di ciò è la "outp (LPT1, TONOON|i)". Per coloro che non conoscono affatto il linguaggio "C", diciamo che l'operatore "|" esegue l'OR tra i due operandi, che nel nostro caso sono il valore TONOON (0x10) e la variabile "i", che assumerà valori diversi compresi tra 0x0

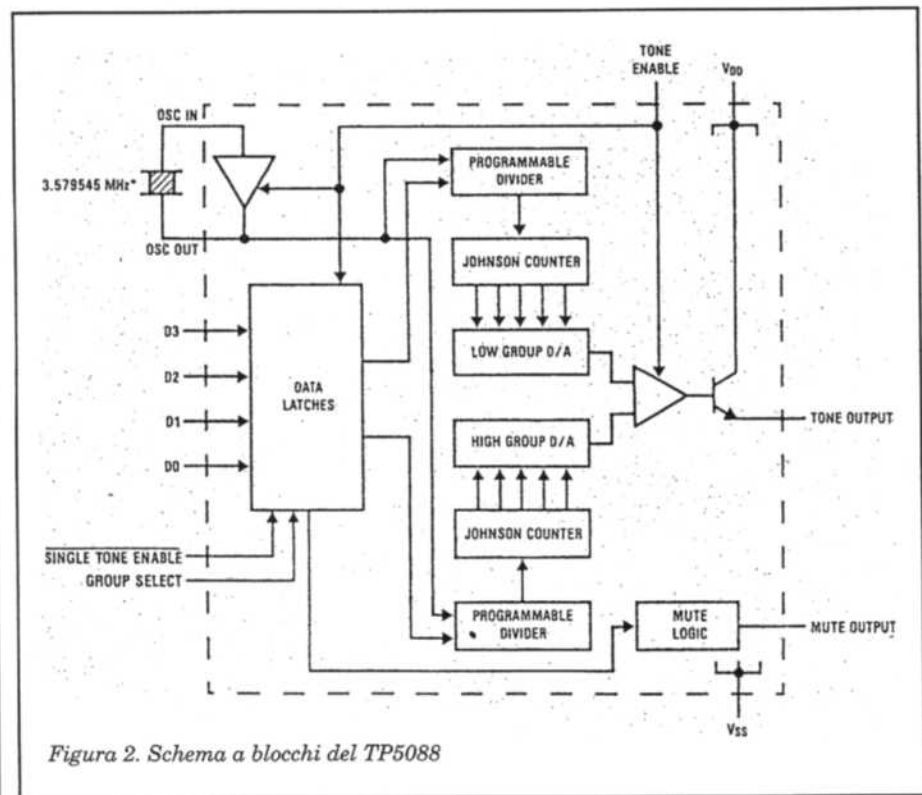
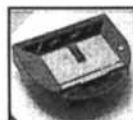


Figura 2. Schema a blocchi del TP5088

sia per avvenimenti più sobri. Scrivendo un programma di poche righe, è possibile implementare giochi di luce che non hanno niente da invidiare a quelli presenti sul mercato e gestiti con un microcontrollore: basta pensare alla sola possibilità di memorizzare su disco rigido o su floppy un numero quasi infinito di programmi oppure alla semplice programmabilità attraverso la tastiera ed il monitor.

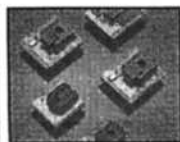
Ma prima di passare alla prossima applicazione, vediamo come avviene l'accensione dei singoli Led in successione: il ciclo è controllato dalla variabile "i" che viene inizializzata con il valore 1. Al valore 1 corrisponde l'accensione del Led numero uno, poiché al valore 1 di un registro equivale avere un 1 al bit meno significativo e zero a tutti gli altri bit. Al contrario del programma precedente, al successivo giro nel loop, la variabile di controllo "i" non viene incrementata di 1, ma viene moltiplicata per 2. Ciò significa che il suo valore successivo sarà 2, con conseguente accensione del Led numero 2. Alla terza iterazione però, tale variabile diventa

PROGRAMMATORE UNIVERSALE ALLO7 (Per PC)



Disponibile in due modelli:
1° Con scheda interna al PC
2° Per la porta parallela
L'ALLO7 programma EPROM - EEPROM - PROM - PAL - Flash - EPROM - MONOCHIP, ecc.

CONVERTITORI



1° Per Programmatori
Sul vostro programmatore, possibilità di programmare: PGA, SOT, PLCC, QFP
2° Per emulatori e test
Possibilità di convertire tutti i tipi di sonda in altro tipo e tutti i tipi di zoccolo (es.: PGA in DIL)

PLD COMPILER



Compiler Jeduc per PLD - FPLD - ecc...
Disponibile la versione Windows

ROM IT



Emulatore di EPROM
Modulo per EPROM da 2764 a 8 Mb
Modulo da 1 a 8 EPROM

EZ - ROUTE DOS :
Disegno di schemi e di SBROGLIATURA AUTOMATICA di circuiti stampati
EZ - ROUTE WDS :
Versione windows di EZ - ROUTE
EASY PC
Disegno di schemi e di SBROGLIATURA AUTOMATICA di circuiti stampati

PROGRAMMATORE per EPROM



Modello DATAMAN - portatile
Modello EP01 copia da 1 fino a 1 Mb
Modello EP02 copia da 4 fino a 1 Mb
Modello SEP01 copia da 1 fino a 4 Mb
Modello SEP04 copia da 4 fino a 4 Mb
Modello MP100 porta seriale 8 Mb - 9751
Modello PIC18

EMULATORE
•
COMPILATORE
•
SCHEDA di Applicazione
•
SIMULATORE
•
ASSEMBLATORE
•

Per :
8031/51
8751/52

87xxx
68HC11
68HC16

6800
6809
68xxx

6502
65816
6805

68705
68HC05
Z80

Z180
H8/300
H8/500

TMSxxx

SVILUPPO di schede con chip



Hardware
Lettore/Programmatore di schede PC BUS, per tutte le versioni di schede
Software
Compiler - Debugger C per PC MS-DOS

PC Interface Protector



- Permette di collegare schede da 8/16 bit al PC senza aprirlo
- Permette il test e la riparazione
- Protetto da fusibili

ANALIZZATORE LOGICO

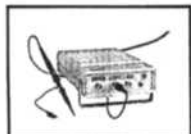


HS 1611
16 Ingressi fino a 100 MHz
HS 3211
32 Ingressi fino a 100 MHz
LA 4240
40 Ingressi fino a 200 MHz
LA 4245
40 Ingressi fino a 400 MHz

Handyprobe (1KHz):
Oscilloscopio + voltmetro + analizzatore di spettro + registratore

Handyscope (40KHz):
Oscilloscopio + voltmetro + analizzatore di spettro + registratore

TP208 (20 MHz):
Oscilloscopio + voltmetro + analizzatore di spettro + registratore



EMULATORE UNIVERSALE ICE V

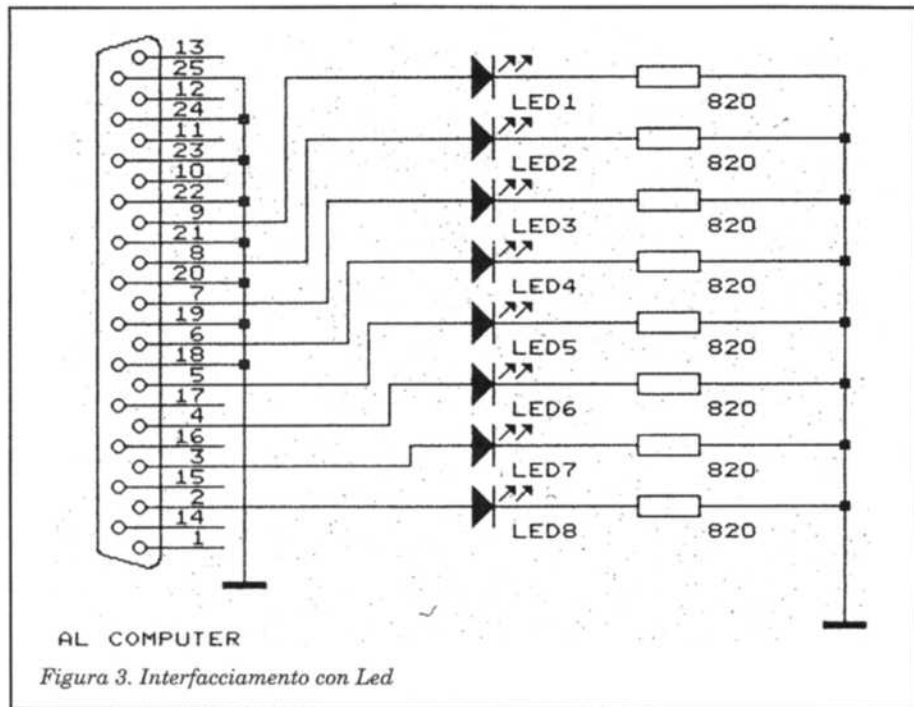


Per :
Z80 - Z180 - 64180 - 68000 - 68010 - 6809 - 6802 - 8088 - 8086 - 80188 - 80C188 - 68HC11 - 8031 - 8051, ect...
altri modelli : PIC16, DSP XXX

SCHEDA DI APPLICAZIONE



Modello per 80C196KB
Modello per Z180
Modello per 80188
Modello per 80C552
Modello per 68HC11
Modello per 68HC16
Modello per 80535
Modello per 803/51/52
Modello per 68322



AL COMPUTER

Figura 3. Interfacciamento con Led

non 3, ma 2×2 ovvero 4, facendo illuminare il Led 3 (perché al 4 corrisponde il numero binario 0000100). E così via fino al bit 7 in cui la variabile varrà 128 ossia $0x80$ che in binario equivale al numero 10000000. Nel linguaggio però, sarebbe stato possibile sfruttare l'opera-

tore di shift " \ll ", ovvero un operatore che consente di shiftare il contenuto di un registro. Abbiamo preferito la nostra soluzione per una portabilità dal "C" ad altri linguaggi come il BASIC, sprovvisti di tale operatore. Con la stessa semplicità con cui vengono gestite le lampade, è

possibile pilotare qualsiasi altro carico. Tanto per fare un esempio, riportiamo ora il programma PULSE.C che consente la composizione di un numero telefonico in modalità decadica. Si intuisce che il relè che controlla la linea telefonica è quello collegato al D0 della porta parallela. Cerchiamo di capire il significato delle istruzioni: la "sprint" inserisce nell'array "numero" la stringa posta tra virgolette e cioè il numero che successivamente verrà composto con le istruzioni "outp(LPT1, LINEAON)" e "outp(LPT1, LINEAOFF)" si chiude e si apre rispettivamente la linea telefonica.

Si noti che dopo la prima chiusura della linea si attendono 2 secondi necessari ad ottenere il segnale di libero dalla centrale. Poi troviamo due loop uno interno all'altro: il primo, controllato dalla variabile "i", cicla per un numero di volte pari al numero di cifre che compongono il numero da chiamare. Il secondo, controllato dalla variabile j, apre e chiude la linea con un rapporto di 60 e 40 ms in funzione della cifra corrispondente alla variabile del loop più esterno. Come vedete, chi lavora da anni solo con linguaggio BASIC, non avrà certamente problemi a programmare anche in "C", poiché a parte le inizializzazioni, la struttura dei programmi è abbastanza simile.

continua

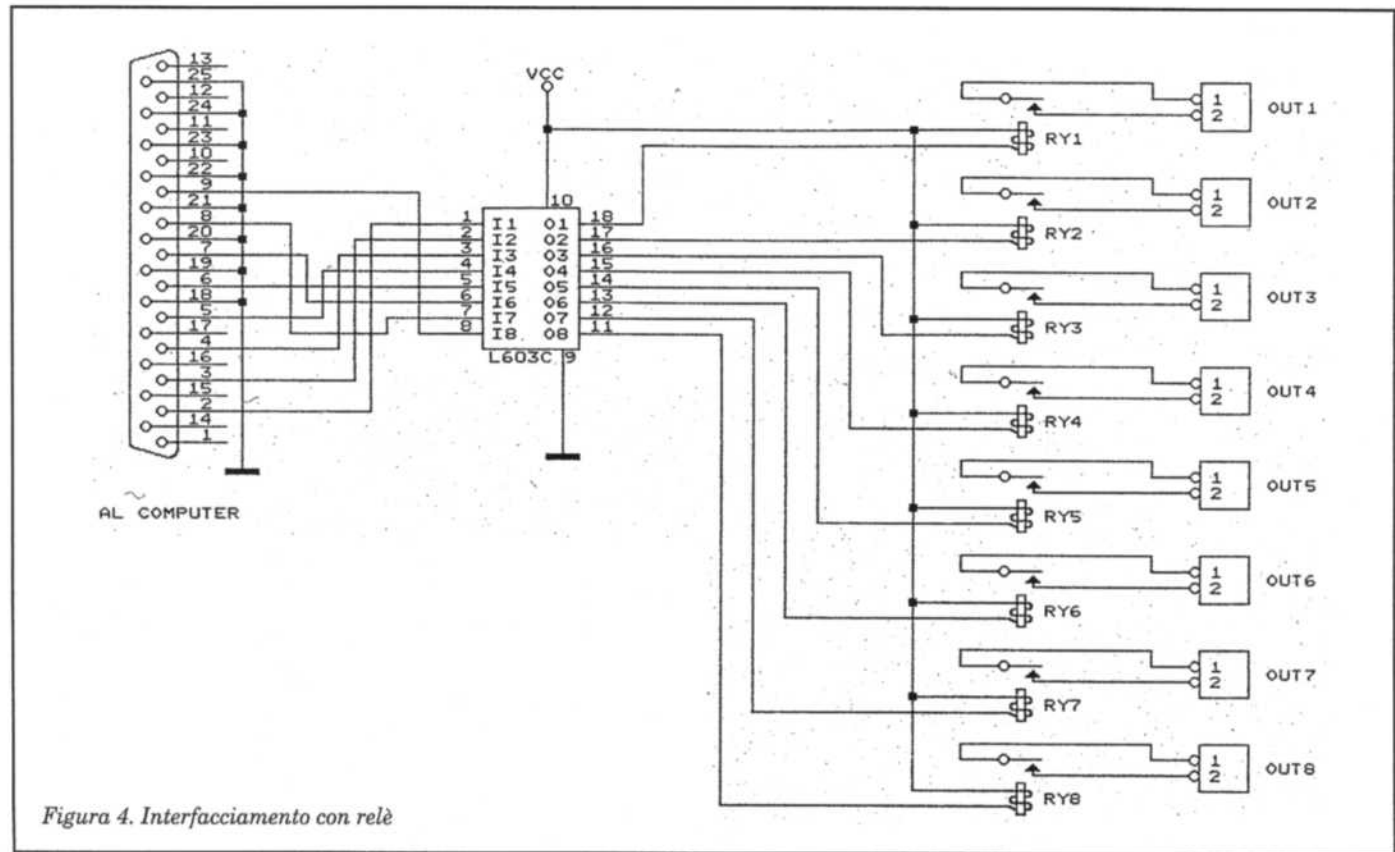


Figura 4. Interfacciamento con relè

IMPARARE A GESTIRE LA PORTA PARALLELA

Da oggi non sarà più un problema sfruttare le potenzialità della porta parallela di un computer, perché vi spiegheremo tutti i meccanismi per usarla nel migliore dei modi

Paolo Sbrana - 2ª parte

Come abbiamo già anticipato nel corso della precedente puntata, con la porta parallela di un personal computer non solo è possibile prelevare dei segnali di uscita, ma è altresì possibile monitorare fino a 4 segnali provenienti dall'esterno.

Per le caratteristiche elettriche della porta stessa, la limitazione è che tali segnali siano TTL compatibili, ovvero con tensione di zero o di cinque volt.

Anche in questo caso le applicazioni possibili sono tantissime e, alcune di queste, anche molto richieste.

Se ricordate, nel mese di aprile ab-

biamo proposto un analizzatore di stati logici implementato proprio sulla porta parallela con soli 4 diodi zener e otto resistenze: il funzionamento era tutto gestito da software, senza particolari accorgimenti hardware, ma nonostante ciò ha riscontrato un notevole successo, probabilmente per il suo costo esiguo.

Con le indicazioni che daremo tra poco, vi sveleremo quali sono le istruzioni base per poter realizzare uno strumento simile, in modo tale da permettere a chiunque di scrivere il "proprio" eseguibile, in funzione delle proprie esigenze di lavoro.

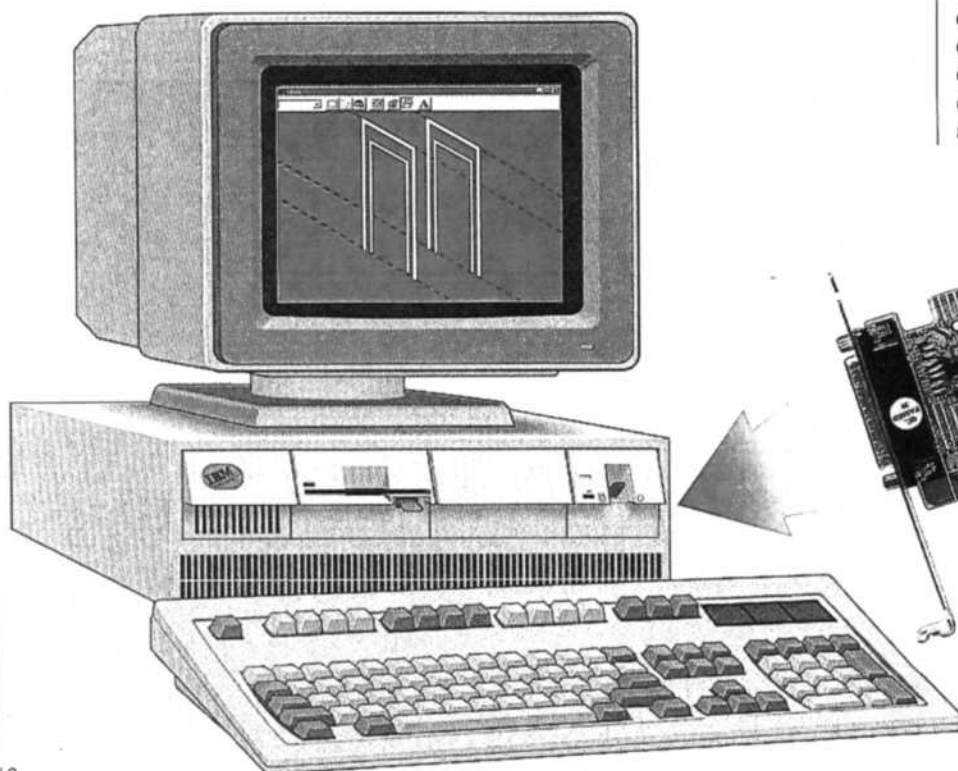
L'acquisizione di un segnale

Passiamo allora a vedere come si può leggere il livello di un segnale presente sulla porta parallela attraverso il software, aiutandoci con il semplice schema di Figura 5: un pulsante è collegato all'ingresso dell'acknowledge (pin 10).

Premendo il pulsante P1, il livello sul pin 10 andrà a 0 volt, rilasciandolo il livello salirà a 5 volt. Vediamo il software di interpretazione (Programma 4).

Prima di proseguire con la spiegazione del software, dobbiamo dire che dalla porta parallela è possibile prelevare lo stato di quattro segnali presenti sui pin 10, 11, 12 e 13. Questi quattro pin corrispondono rispettivamente ai segnali ACKNOWLEDGE, BUSY, OUT OF PAPER e SELECT presenti su quattro bit di un registro di stato della porta e quindi interrogabile da software. Di questi quattro segnali, due sono "diritti" e due "rovesciati", ovvero il loro stato è mappato sul registro di stato in modo diverso: i pin 10 e 11 se collegati a massa, settano i bit corrispondenti del registro di stato ad "1", mentre i pin 12 e 13 li settano a zero.

Il pin che noi abbiamo utilizzato per questa prova è del tipo "rovesciato", quindi quando leggeremo il suo bit a "0", vorrà dire che su quel pin è presente un segnale di 5 volt, viceversa quando leggeremo il suo bit a "1", il segnale sarà di zero volt.



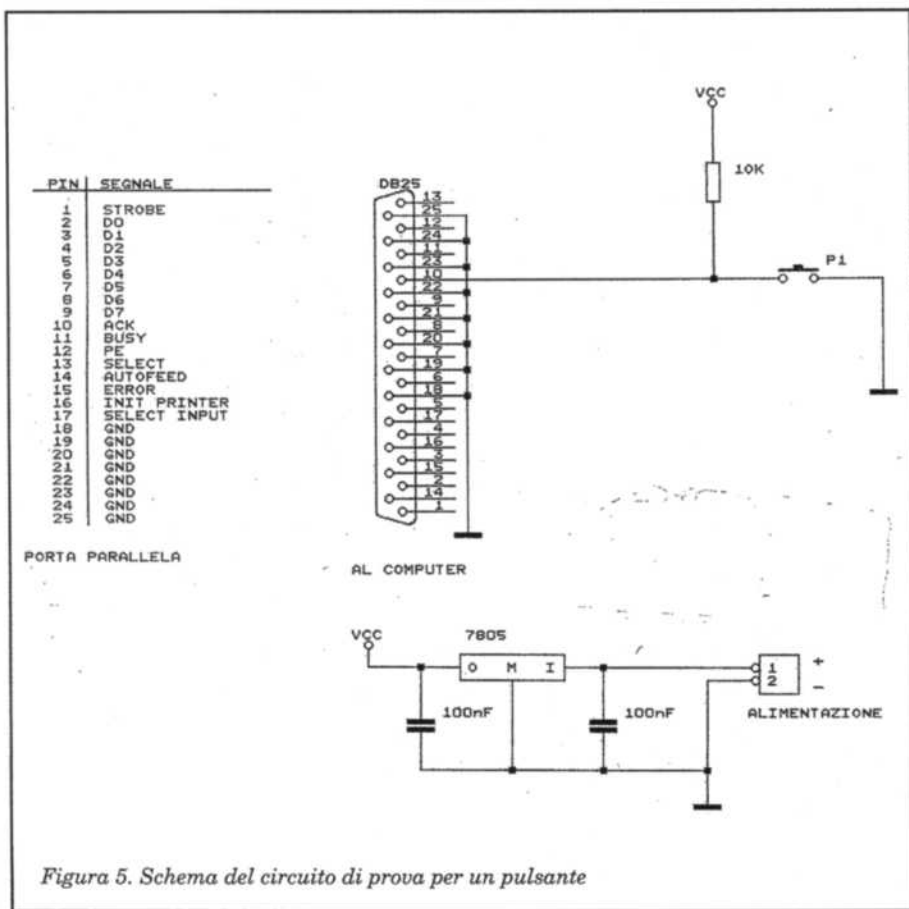


Figura 5. Schema del circuito di prova per un pulsante

Detto questo, analizziamo dettagliatamente il programma. Questa volta la porta parallela non è più indirizzata con 0x378, ma viene definita con PORTNUM = 0. Se la parallela era la LPT2, PORTNUM andava definito con 1. È stata inserita la variabile "fineciclo" che permette di uscire dal programma una volta entrato in esecuzione.

Il corpo del programma è un ciclo do-while, da cui si esce quando la variabile "fineciclo" diventa diversa da zero.

Per far ciò, abbiamo introdotto il test sulla pressione di un tasto sulla tastiera con l'istruzione dedicata "kbhit()".

Quando un tasto viene premuto, si va ad analizzare il valore relativo al tasto letto e si confronta con quello del tasto ESCAPE. Se sono uguali, si pone a "1" la variabile "fineciclo" ed al prossimo ciclo si uscirà dal loop do-while. L'istruzione che va a leggere il registro di stato relativo alla porta parallela è la "biosprint".

Si nota che sono necessari tre parametri: il primo indica l'operazione su tale registro (nel nostro caso lettura dello stato che equivale a 2), il secondo indica il valore da scrivere sulla porta (che a noi non serve) ed il terzo indica il numero della porta (0=LPT1, 1=LPT2, ecc).

Il valore che la funzione biosprint

ritorna, è un byte che identifica lo stato della porta. A noi interessano in particolare modo quattro stati:

- 0x10 (select),
- 0x20 (out of paper),
- 0x40 (acknowledge) e
- 0x80 (busy).

Nel programma appena visto, si fa il confronto dello stato con 0x40, ovvero con il segnale presente sul pin 10 relativo all'ACK. Se tale valore è 0, sul pin ci sono 5 volt e quindi scrivo "1", viceversa scrivo "0". In particolare, se avessi sfruttato il pin 13, avremmo dovuto scrivere:

```
if (valore&0x10)
    printf("1");
else
    printf("0");
```

Notare che è stato tolto l'operatore di negazione (!) ed il confronto è stato fatto con 0x10.

Altri tipi di interfaccia

Se le condizioni di interfacciamento fossero critiche (segnali con forte rumore, sbalzi di tensione, ecc) è possibile adottare una soluzione tipo quella riportata in Figura 7: la separazione tra la porta e i segnali è di tipo ottico.

PROGRAMMA 4

```
/*#####*/
/* ACQUISIZIONE DI UN PULSANTE DALLA PARALLELA */
/*#####*/
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

#define PORTNUM 0 /* parallela */
#define STATUS 2
#define ESC 0x1b
#define ACK 0x40

typedef unsigned char BYTE;

int main(int argc, char *argv[])
{
    int valore;
    int value = 0;
    int fineciclo = 0;

    do
    {
        if(kbhit())
        {
            if(valore==(BYTE)(_bios_keybrd(_KEYBRD_READ))==ESC)
                fineciclo = 1;
        }
        valore=(biosprint(STATUS, value, PORTNUM));
        if(!valore&ACK)
            printf("1");
        else
            printf("0");
    }
    while(fineciclo == 0);
    return 0;
}
```

PROGRAMMA 5

```

/
/*****
/ CONTROLLO ON/OFF PER PULSANTI VIA SOFTWARE */
/
/*****
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

#define PORTNUM 0 /* parallela */
#define STATUS 2
#define ESC 0x1b

typedef unsigned char BYTE;

int main(int argc, char *argv[])
{
    int valore, fineciclo;
    int valore = 0;
    int memory1, memory2, memory3, memory4 = 0;

    fineciclo = 0;
    do
    {
        if(kbhit())
        {
            if(valore=(BYTE)(_bios_keybrd(_KEYBRD_READ))==ESC)
                fineciclo = 1;
        }
        valore=(biosprint(STATUS, valore, PORTNUM));
        if(valore&0x10)
        {
            if(memory1==0)
            {
                printf("1on ");
                memory1 = 1;
            }
        }
        else
        {
            if(memory1==1);
            {
                printf("1off ");
                memory1 = 0;
            }
        }
        if(valore&0x20)
        {
            if(memory2==0)
            {
                printf("2on ");
                memory2 = 1;
            }
        }
        else
        {
            if(memory2==1);
            {
                printf("2off ");
                memory2 = 0;
            }
        }
        if(!((valore&0x40)))
        {
            if(memory3==0)
            {
                printf("3on ");
                memory3 = 1;
            }
        }
        else
        {
            if(memory3==1);
            {
                printf("3off ");
                memory3 = 0;
            }
        }
        if(!((valore&0x80)))
        {
            if(memory4==0)
            {
                printf("4on ");
                memory4 = 1;
            }
        }
        else
        {
            if(memory4==1);
            {
                printf("4off ");
                memory4 = 0;
            }
        }
    }
    while(fineciclo == 0);
    return 0;
}

```

Le resistenze di pull-up per la porta sono collegate agli 8 pin relativi all'uscita della parallela, quindi, per vedere funzionare il tutto sarà necessario come prima istruzione inserire la "outp(LPT1,0xff)", che porta ad alto livello tutti questi pin.

Gli optoisolatori devono avere masse separate ed essere correttamente pilotati. Per questo motivo abbiamo inserito l'L603C. Se, malauguratamente, ci fosse qualche extratensione, brucerebbe al massimo tale integrato o, al limite, i quattro optoisolatori, sicuramente fatto più conveniente della rottura della porta stessa, che da qualche tempo è di serie direttamente sulla main-board del computer. Per vedere meglio come con il software sia possibile implementare fa-

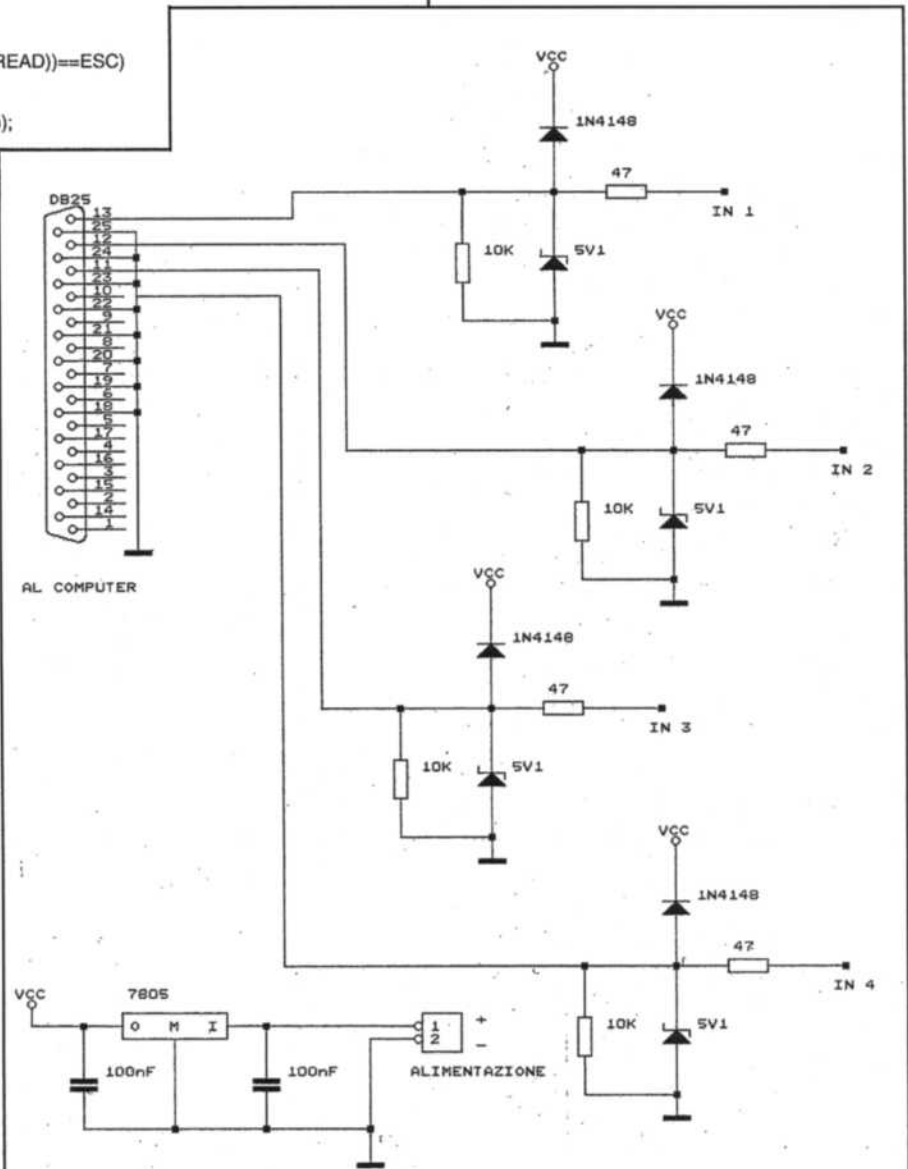


Figura 6. Protezioni per la sicurezza della porta

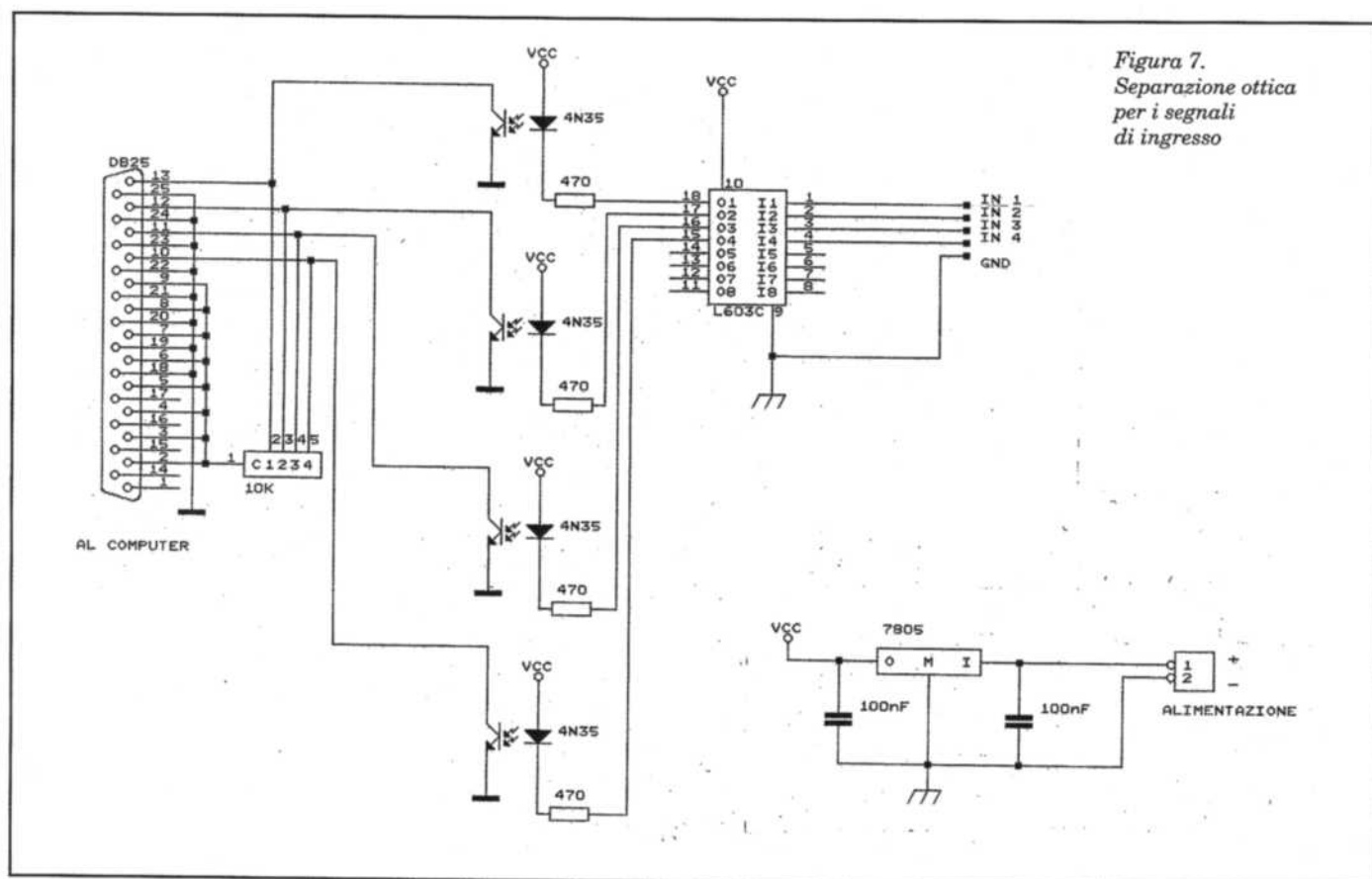


Figura 7.
Separazione ottica
per i segnali
di ingresso

cilmente molte funzioni hardware, torniamo allo schema di Figura 6 e supponiamo di connettere ai quattro ingressi altrettanti pulsanti.

Al contrario di quanto si faceva nel programma precedente, adesso dobbiamo far in modo che siano visualizzati solo variazioni di stato, ovvero lo stesso che implementare un controllo di on/off software (Programma 5).

Con quest'ultimo programma, vengono visualizzati all'inizio gli stati relativi ai quattro pulsanti, poi soltanto le variazioni (i fronti di salita e discesa) indicate con il nuovo stato del canale relativo. Ad esempio, all'inizio potremmo avere la seguente visualizzazione: "1on 2on 3on 4off". Poi, quando un ingresso (per esempio il 3) varia, verrà accodato il nuovo stato per tale ingresso (nel nostro caso "3off").

Anche se ciò potrebbe non sembrare significativo, rappresenta il cuore del software presentato pochi mesi fa dell'analizzatore di stati logici.

Ovviamente, per scrivere un programma completo è indispensabile almeno una minima conoscenza del linguaggio, ma per le istruzioni principali tutti i linguaggi si somigliano: le iterazioni del BASIC vengono eseguite principalmente con l'istruzione FOR, in "C" con FOR

o con DO-WHILE o più semplicemente con WHILE, lo stesso dicasi per il PASCAL. L'unica cosa a cambiare è la sintassi, ma per questo è sufficiente avere a disposizione un manuale. Allo stesso modo, le decisioni vengono prin-

cialmente prese con degli "if" in quasi tutti i linguaggi ad alto livello.

La maggior difficoltà, invece, nel passare da un linguaggio all'altro è invece la grafica: ognuno la gestisce in modo diverso.

Gps per auto, barche, escursioni.

Obbiettivi e misurazioni precise.

Misuratori gas, radiazioni e altre novità U.S.A.

Metal Detectors

ed equipaggiamenti

U.S.A. per ricerca,

industria, security.

I più potenti

in commercio l'hobby

che da soddisfazioni!

Difesa elettronica

e sistemi di sicurezza per proteggere la casa,

l'auto e la famiglia. Visori notturni, ricetrasmittitori

e scanner.

Catalogo gratuito - Zone libere per agenti

Electronics Company

Via Pediano, 3A - 40026 Imola - Tel. 0542/600108



