

Electronica In

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

117

**Il PC diventa
oscilloscopio**

**Interfaccia GSM
Bluetooth**

**Radiocomando
433,92 MHz a 4 CH**

Modulo Real Time Clock universale

Protocollo seriale Velbus: dalla teoria alla pratica

Localizzatore Mambo, funzioni e programmazione

ESCLUSIVO
CORSO LCD
(Liquid Crystal Display)

Modulo Real Time Clock universale

di *Davide Scullino*

Con un ingombro di pochi centimetri quadri e un consumo di corrente irrisorio, genera data e ora di sistema. Facilmente implementabile in qualsiasi apparato digitale, comunica con l'esterno tramite una porta seriale e dispone anche di un segnale di clock programmabile. Grazie ad un sistema di back-up interno, può mantenere i dati in memoria anche in assenza di corrente.



E sistono numerose applicazioni, sia in campo digitale che analogico, in cui serve un orologio di sistema: apparati come controlli accessi, cartellini orari elettronici a transponder o badge magnetico, centralini telefonici, impianti di controllo e gestione delle riprese video, etichettatrici ed altro ancora, normalmente necessitano di qualcosa che, all'occorrenza, comunichi l'orario e la data, per sovrapporli a un'immagine, registrarli accanto al nome del prodotto (sistemi di etichettatura post-produzione) o a una chiamata effettuata da un'azienda o da un albergo, al record inerente al passaggio da un

tornello di una persona autorizzata o meno. Nei sistemi gestiti da un microprocessore o microcontrollore, l'orologio di sistema può essere realizzato con un'apposita routine, che, mediante un timer, fa avanzare secondi, minuti, ore, giorni, mesi ed eventualmente anni, quindi scrive il risultato aggiornato del conteggio in una RAM protetta da una batteria tampone o in una EEPROM. Resta però un problema: quando la gestione di un apparato è affidata a un microcontrollore relativamente semplice, ad esempio ad 8 bit, può accadere che aggiungere ulteriore carico di lavoro a una CPU già oberata sia davvero ➤

Si tratta di un contatore BCD integrato in un contenitore DIP a 8 piedini, che permette di contare ore, minuti, secondi, giorni, mesi e anni; è realizzato in tecnologia CMOS a bassissimo consumo, così da poter funzionare anche con una pila a bottone, della quale deve essere provvisto; il contatore è completato da una static-RAM non volatile da 56 byte, nella quale vengono collocate le informazioni su ora e data, informazioni che poi l'integrato rende disponibili su richiesta dell'apparato che dialoga con esso. Il chip può esprimere l'ora nel formato a 12 (con indicazione AM/PM) o 24

6; le unità di minuti sui bit 0, 1, 2, 3.

Terzo byte (0x02): contiene le ore. Il bit 6 è riservato all'indicazione del formato: 24 o 12 ore. In modalità "24 ore" il bit 4 dà la prima decina di ore (10÷19) e il 5 la seconda decina (20÷23). In modalità "12 ore" il bit 4 ha lo stesso significato del caso precedente, ma il 5 dà l'indicazione AM/PM. I bit 0÷3 contengono le unità delle ore.

Quarto byte (0x03): contiene il giorno, collocato nei bit 0, 1, 2; si tratta del giorno della settimana, espresso con un numero che va da 01 a 07. Il dato corrispondente va interpretato dal dispositivo che riceve i dati dal DS1307, considerando che 01 è lunedì e 07 domenica.

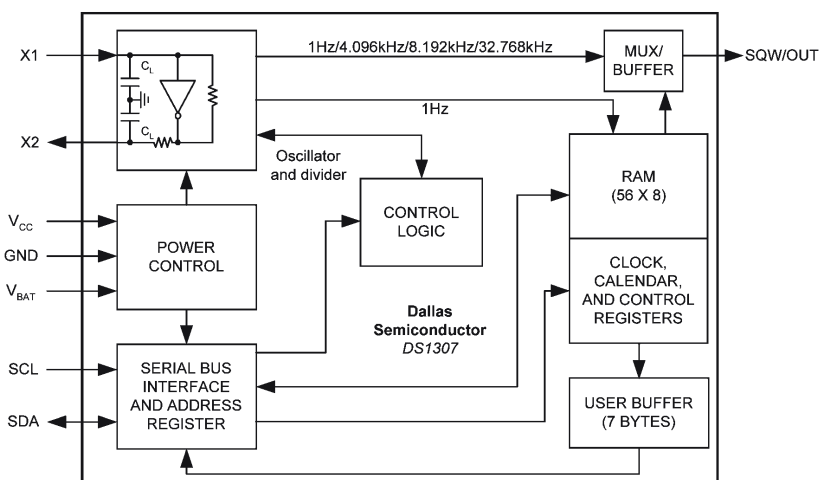
Quinto byte (0x04): contiene la data, ossia il numero del giorno del mese (01÷31) suddiviso in decine (bit 4 e 5) e unità (bit 0, 1, 2, 3) di giorni.

Sesto byte (0x05): contiene il mese (01÷12) scomposto in decine (bit 4, mesi dal 10 a 12) e unità (bit 0, 1, 2, 3, ossia mesi dal 01 al 09).

Settimo byte (0x06): contiene le informazioni sull'anno, nel formato a due cifre (00÷99); nei bit 0, 1, 2, 3 si trovano le unità e nei 4, 5, 6, 7 le decine.

La sessione di comunicazione termina con l'invio della condizione di stop.

L'uscita SQW è inattiva se il bit 4 (SQWE) del registro di controllo vale 0; in tal caso si può comandare la condizione logica impostando opportunamente il bit 7 (OUT) del solito registro: l'uscita si pone ad 1 logico se il predetto bit 7 è ad 1 e a zero se, invece, esso vale 0 logico.



ore, e provvede automaticamente al passaggio da un mese al seguente riconoscendo i mesi con meno di 31 giorni; effettua altresì la correzione per l'anno bisestile.

Incorpora anche un gestore dell'alimentazione in grado di rilevare l'abbassamento sotto un certo livello della tensione principale (applicata al pin 8, Vcc): in questo caso, il controllore prende tensione dalla batteria (collegata tra il piedino 3, Vbat e il pin 4, massa) per mantenere attivi i contatori e la memoria e spegne i circuiti ad alto consumo (porta di comunicazione, uscita clock...).

La comunicazione con il microprocessore avviene mediante un I²C-bus, disponibile sui piedini 5 (SDA, ossia canale dati) e 6 (SCL, cioè il clock); il dispositivo si comporta come uno Slave, quindi può comunicare solamente quando gli è richiesto. La comunicazione in modalità Slave avviene inviando una condizione di start dal dispositivo che la inizia, quindi l'indirizzo identificativo della periferica, che inizialmente è 00 (zero); prosegue comunicando i dati degli altri registri, tra i quali spicca quello di controllo dell'uscita SWQ, dalla quale è possibile prelevare un segnale di clock da portare a dispositivi che possono averne bisogno. Si tratta di un'onda quadra la cui frequenza dipende dallo stato del registro di controllo, ossia dallo stato dei suoi primi due bit (RS0 e RS1); partendo dalla frequenza di clock di base ricavata dal quarzo a 32,768 MHz, l'uscita SQW fornisce:

- 1 Hz, se i bit sono entrambi zero;
- 4,096 kHz se RS0 vale 1 e RS1 zero;
- 8,192 kHz se RS0 vale 0 ed RS1 uno;
- l'intero clock (32,768 kHz) quando i bit valgono entrambi 1.

Dopo l'inizio di una sessione di comunicazione e l'invio del byte contenente lo stato del registro di controllo, il DS1307 risponde inviando le informazioni riguardanti orologio e calendario (Timekeeper Registers) tramite una stringa composta da più byte, contenenti ciascuno uno dei parametri, ossia secondi, minuti, ore, giorno, mese, anno.

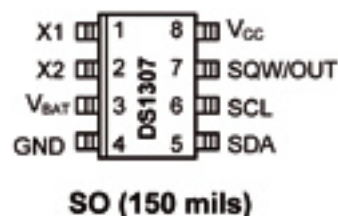
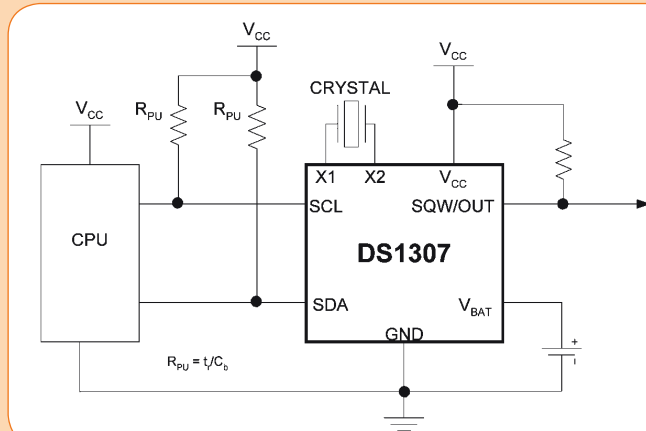
La stringa è composta dai seguenti byte:

Primo byte (0x00): contiene i secondi. Le decine di secondi si trovano sui bit 4, 5, 6; unità di secondi sono invece sui bit 0, 1, 2, 3.

Secondo byte (0x01): contiene i minuti. Le decine di minuti sono sui bit 4, 5,

I piedini dell'integrato DS1307 sono i seguenti:

- 1 = X1; pin di collegamento del quarzo dell'oscillatore;
- 2 = X2; pin di collegamento del quarzo dell'oscillatore;
- 3 = Vbat; positivo della pila di mantenimento (backup);
- 4 = GND; massa comune dell'alimentazione principale e della pila;
- 5 = SDA; Serial Data, ossia canale dati dell'I²C-bus;
- 6 = SCL; Serial Clock, cioè clock dell'I²C-bus;
- 7 = SQW/OUT; uscita da cui prelevare l'onda quadra;
- 8 = Vcc; alimentazione principale (5 V).



troppo; senza contare che una routine di orologio può arrivare a portare via 1 kbit di memoria di programma e ciò obbliga a sacrificare altre funzioni anche in un micro relativamente capiente come, ad esempio, il PIC16F877, che conta su ben 8k di Flash. Dunque, per tutti i progettisti che vogliono “spremere” al massimo la CPU del proprio sistema, abbiamo pensato di realizzare e pubblicare un modulo RTC (Real Time Clock) che lavori autonomamente e che sollevi il microcontrollore o microprocessore del caso dall'onere di doversi occupare della gestione dell'orologio di macchina.

Il progetto qui descritto è un modulo di dimensioni tanto ridotte da poter essere inserito in qualsiasi apparato o su qualunque scheda a microprocessore; esso dialoga mediante pochissime linee (ha dieci piedini in tutto) e può essere configurato in maniera da fornire le informazioni su data e ora sia su interrogazione del sistema nel quale viene inserito, sia periodicamente in modo automatico; il dispositivo può inoltre essere programmato per impostare il formato di data e ora preferito dal sistema nonché altri parametri di funzionamento.

Per agevolare la programmazione e la connessione del modulo con gli apparati che devono utilizzarlo, abbiamo pensato di realizzare una piccola scheda di supporto, che oltre a fare da base e ancoraggio, permette di portare all'esterno, mediante un connettore RJ45, le linee più importanti (il bus I²C di comunicazione con il programmatore, la tensione di programmazione V_{pp} e il canale TX/RX per il dialogo con l'esterno) oltre ai contatti di alimentazione. Inoltre, per dare la possibilità di configurare il modulo mediante un PC provvisto di porta USB, abbiamo studiato e pubblichiamo in queste pagine un'interfaccia seriale/USB da collegare al connettore RJ45 della basetta di supporto; tale interfaccia

dispone altresì di un alimentatore stabilizzato e uno strip per consentire la programmazione in-circuit, in alternativa a quella da PC.

Il modulo RTC

Ma procediamo con ordine ed esaminiamo innanzitutto il “nocciolo” della questione, ossia il piccolo modulo orologio: è basato su un chip della Maxim-Dallas siglato DS1307, il quale comunica con un piccolo microcontrollore Microchip a otto piedini tramite una linea I²C-bus.

Il dispositivo, per comunicare con i sistemi esterni, necessita di dieci linee tra le quali un'interfaccia dati seriale bidirezionale a livello TTL (TX/RX) e un I²C-bus per la programmazione In-Circuit.

La linea dati fa capo al PIC montato nel nostro modulo RTC e così pure l'I²C-bus che va all'esterno, da non confondere con quello del DS1307 che comunica esclusivamente con il PIC. Il DS1307 è sostanzialmente un contatore BCD (Binary Coded Decimal) a basso consumo, che conta secondi, minuti, ore, giorni, mesi e anni, provvisto di 56 byte di RAM statica non volatile; il componente provvede automaticamente a determinare quali sono i mesi con meno di 31 giorni e ad effettuare la correzione per l'anno bisestile (che comporta il 29° giorno a febbraio). L'orologio può operare nelle modalità 12 o 24 ore, con indicazione delle ore antimeridiane (AM) e di quelle pomeridiane (PM).

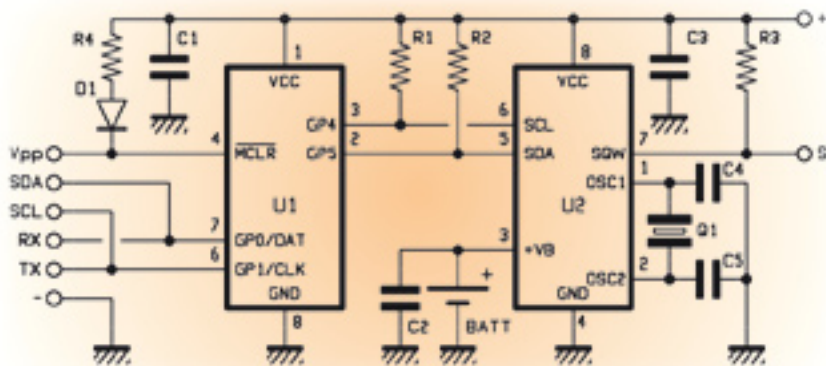
Le informazioni sull'ora e la data vengono collocate in un registro e trasferite (mediante l'I²C-bus del quale il chip è provvisto) dietro interrogazione al microcontrollore, il quale poi, in base a come viene configurato, le invia all'esterno periodicamente (in modo automatico) o su specifica richiesta. Oltre a quanto detto, il DS1307 dispone di un'uscita di clock programmabile: più esatta-

mente, rende disponibile un'onda quadra ricavata dalla frequenza di clock dell'orologio (determinata, a sua volta, dal quarzo da 32,768 kHz collegato ai piedini 1 e 2) che, mediante un apposito divisore interno, può essere divisa di frequenza ottenendo 1 Hz, 4,096 kHz, 8,192 kHz o l'intero clock. Le frequenze ottenibili non sono state scelte a caso: per esempio, 1 Hz può servire a far lampeggiare i due punti o il punto dei secondi dell'eventuale display che mostra l'ora. La condizione dell'uscita di clock ausiliario (SQWE, piedino 7) si definisce impostando opportunamente lo stato logico dei bit RS0 (0) ed RS1 (1) del registro di controllo, secondo quanto mostrato nell'apposito riquadro; ad esempio, 1 Hz si ottiene con entrambi i bit a zero. Si noti che quando sia il quarto (SQWE) che il settimo bit (OUT) si trovano a zero logico, l'uscita di clock si pone fissa a livello basso; se, invece, il bit 7 è ad uno logico e il 4 a zero, l'uscita assume costantemente lo stato alto.

Il chip dispone anche di un circuito di controllo in grado di verificare la mancanza dell'alimentazione principale (V_{cc}) e fare in modo che la tensione occorrente venga prelevata dalla batteria di backup, nel nostro caso una pila a bottone da 3 V, che deve essere collegata tra il piedino 3 (V_{bat}) e la massa di alimentazione (4,



SCHEMA ELETTRICO



ELENCO

COMPONENTI:

R1: 4,7 kohm SMD
R2: 4,7 kohm SMD
R3: 4,7 kohm SMD
R4: 1 kohm SMD
C1: 100 nF
multistrato SMD
C2: 100 nF
multistrato SMD
C3: 100 nF
multistrato SMD
C4: 15 pF



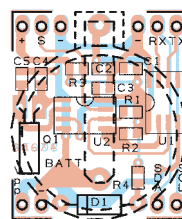
ceramico SMD

C5: 15 pF ceramico SMD
D1: 1N4007
U1: PIC12F675-SO (MF674)
U2: DS1307-SO
Q1: quarzo 32.768 kHz



Varie:

- Porta batteria CR2032
- Batteria CR2032
- Strip maschio 2 pin (2 pz.)
- Strip maschio 3 pin (2 pz.)
- circuito stampato S674



ossia GND). La sezione di controllo dell'alimentazione è dimensionata in modo da svolgere due compiti: preservare la memoria in cui sono contenuti i dati sull'ora e la data attuali e mantenere attivo il contatore dell'orologio. Il primo viene svolto conservando nella RAM non volatile le informazioni corrispondenti, mentre il secondo è espletato dall'alimentazione di riserva presa dalla pila o batteria tampone collegata fra il piedino 3 e il 4. Oltre a ciò, va detto che la sezione di controllo dell'alimentazione interviene sulle altre funzioni del DS1307, bloccando la comunicazione con l'I²C-bus e l'emissione dell'eventuale segnale di clock ausiliario dal piedino 7; il tutto, allo scopo di minimizzare il consumo di energia ed estendere quanto più possibile l'autonomia, la quale, comunque, può arrivare ad alcuni mesi. Si noti che l'interven-

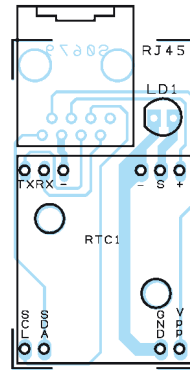
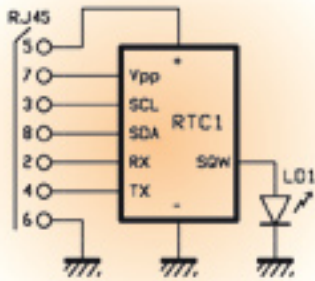
to del blocco di controllo si verifica quando la tensione letta tra il piedino dell'alimentazione principale (8) e massa (4) è inferiore al prodotto $1,25 \times V_{bat}$, ossia diviene minore di 1,25 volte la tensione della pila.

Bene, dopo aver spiegato in maniera essenziale come funziona il DS1307, dobbiamo dire che il microcontrollore inserito nel modulo RTC serve essenzialmente ad interfacciare tale chip con l'esterno, almeno per quel che riguarda due funzioni: l'impostazione del formato dei dati in uscita, o programmazione, e la modalità di comunicazione.

Nella modalità programmazione, possiamo istruire il micro su come deve trattare e inviare all'esterno i dati ricevuti dall'orologio; comprendete ciò rammentando che il DS1307 comunica le informazioni del caso solo quando gli è richiesto

con apposite istruzioni, cosa che, nel nostro caso, viene compiuta proprio dal PIC. Richiesta e invio avvengono sull'I²C-bus del chip Maxim-Dallas. Ora, se si desidera che il modulo RTC comunichi data e ora di sistema automaticamente, si deve programmare il micro allo scopo; in tale situazione il PIC interroga il DS1307 ad intervalli prestabiliti, acquisisce i dati corrispondenti e li invia all'esterno mediante la linea dati TX/RX. Invece, se si vuole che il modulo mandi le informazioni solo quando gli viene richiesto dall'esterno (sempre mediante la linea dati seriale TX/RX) occorre programmarlo di conseguenza; in tal caso, all'arrivo di una richiesta il PIC si fa interprete dell'istanza e la trasmette al DS1307 usando l'I²C-bus di cui il chip è provvisto. Quest'ultimo risponde inviando al microcontrollore (sempre lungo il suo bus I²C) la risposta e le informazioni su ora e data, il quale li rimanda al dispositivo esterno che ha inviato la richiesta. Sempre a proposito del micro, va detto che dispone di due linee di comunicazione: una I²C usata per comunicare con l'orologio, ed una seriale, che, a seconda dei casi, è sincrona (SDA ed SCL) o asincrona (TX ed RX). Viene usata in modalità sincrona quando si trasferisce il firmware nel micro per mezzo di un programmatore; quando, invece, il micro è programmato e in funzione, essa si comporta come seriale classica che, all'occorrenza e mediante un'apposito convertitore TTL/RS232, può comunicare con un PC. Per interrogare e impostare il modulo RTC dall'esterno è necessario utilizzare il semplice protocollo di comunicazione descritto nella tabella di pagina 21. Il canale seriale funziona in half-duplex ed è impostato per una velocità di comunicazione di 9.600 bps, 8 bit di dati, nessuna parità e 1 bit di stop; serve tanto per impartire i comandi di programmazione e interrogazione

SCHEMA ELETTRICO



ELENCO COMPONENTI:

LD1: led 3 mm rosso

Varie:

- Strip femmina 2 pin (2 pz.)
- Strip femmina 3 pin (2 pz.)
- Connettore RJ45
- Circuito stampato codice S679

dell'orologio, quanto per ricevere da questo ora e data. Il modulo RTC, che deve essere alimentato con una tensione continua ben stabilizzata del valore di 5 volt, assorbe una corrente (davvero esigua) dell'ordine di 1,5 milliampere, che scende a 500 nA nel funzionamento a pila.

La scheda di adattamento

Come anticipato, lo scopo dell'RTC è quello di aggiungere un orologio ad una piattaforma a microprocessore, in maniera facile e con limitato consumo di risorse. Esso è realizzato sotto forma di modulo con una serie di pin a passo 2,54 mm per le connessioni alla scheda micro. Talvolta però è necessario collegarlo ad un computer per effettuare la programmazione ed eventuali test, perciò abbiamo realizzato una basetta di adattamento che rende disponibili le linee di ingresso e uscita su un connettore RJ45. Su questa basetta è presente anche un led, che segnala l'attività dell'uscita ausiliaria di clock e che risulta acceso a luce fissa se quest'ultima è disattivata e posta a 1 logico, spento se la stessa è disattivata ma forzata a zero, pulsante alla frequenza di clock scelta se SQW/OUT è attiva. La basetta va considerata come interfaccia per test di laboratorio, non indispensabile per il funzionamento dell'orologio, ma può anche essere utilizzata come accessorio per facilitare la connes-

sione a sistemi finiti e inscatolati, potenzialmente dotati di connessioni USB. In questo caso il modulo di adattamento da USB a seriale che abbiamo preparato è un altro accessorio molto utile, se non necessario.

Il modulo convertitore

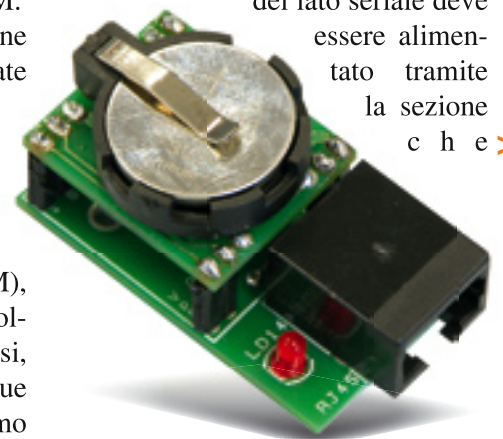
Si tratta di un circuito di conversione da USB a RS232 con livelli TTL, il cui schema elettrico è riportato nella pagina seguente.

Dando uno sguardo allo schema, si nota che sul connettore RJ45 sono disponibili tutti i segnali presenti sul connettore RJ45 della scheda di adattamento. Da questa presa, i vari segnali prendono due direzioni distinte: le linee SCL/SDA (che, ricordatelo, sono quelle del PIC e non del DS1307), la massa e la linea della tensione Vpp vengono portate ad un connettore pin-strip, da usare per la programmazione in-circuit del PIC. Le linee TX ed RX, invece, sono collegate al chip U1, un FT232BM. Le restanti linee (alimentazione +5V e massa) vengono prelevate direttamente dalla porta USB del computer e sono comuni a tutti i tre circuiti (orologio, scheda di adattamento e convertitore USB).

Poiché l'integrato U1 (FT232BM), il convertitore vero e proprio, svolge da solo molti compiti complessi, il circuito elettrico che ne consegue è decisamente semplice. Riteniamo

quindi utile fornirvi qualche ulteriore informazione riguardo a tale chip, qualora lo vogliate utilizzare in applicazioni differenti dalla nostra. L'FT232BM della FTDI (www.ftdichip.com) può essere utilizzato sia "stand-alone" prelevando l'alimentazione dalla porta USB (come nel nostro caso) sia all'interno di un altro circuito quale, ad esempio, un sistema a microprocessore; in quest'ultimo caso, il chip può essere alimentato dallo stesso circuito che alimenta il micro. Sempre in tema di alimentazione, va detto che il componente prevede due pin distinti per la sezione di I/O (verso la linea seriale TTL) e quella di conversione USB. Quando tutto il convertitore funziona con i 5 V prelevati dall'USB del computer, i due pin vanno solitamente uniti; se invece si desidera distinguere la sezione di I/O da quella USB, il positivo del lato USB va alimentato dal PC e quello

del lato seriale deve essere alimentato tramite la sezione che



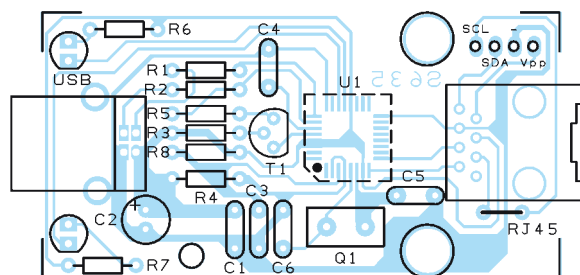
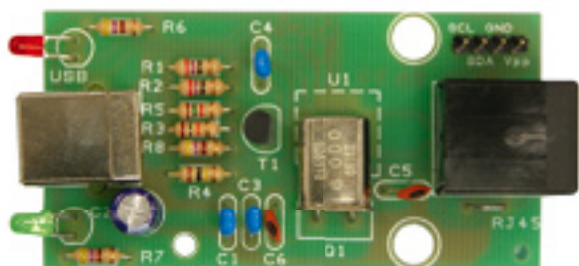
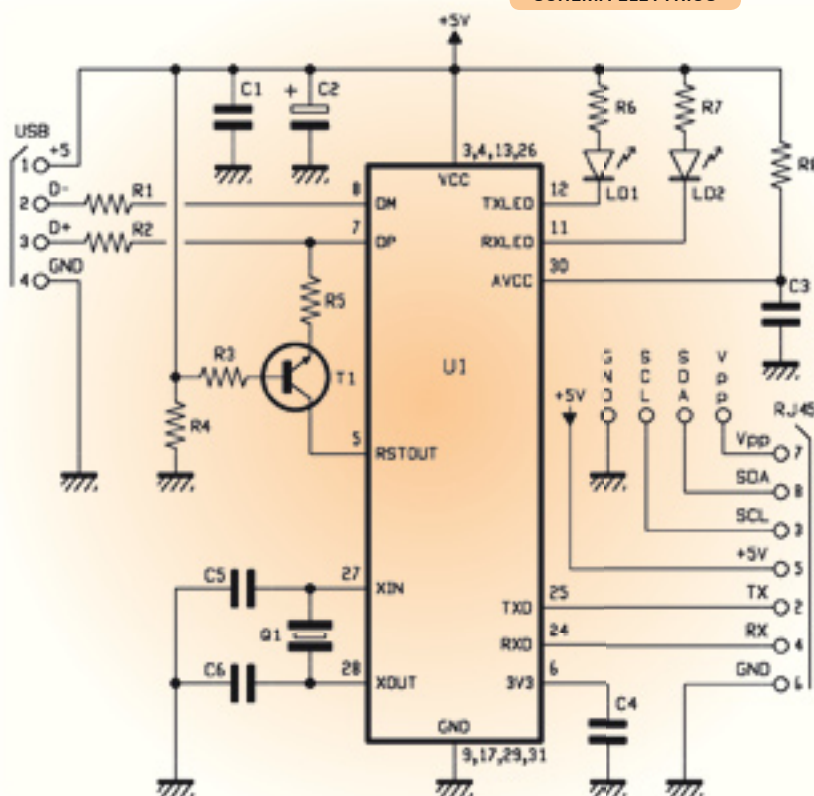
ELENCO COMPONENTI:

R1: 27 ohm
R2: 27 ohm
R3: 10 kohm
R4: 100 kohm
R5: 1,5 kohm
R6: 470 ohm
R7: 470 ohm
R8: 470 ohm
C1: 100 nF multistrato
C2: 220 μ F 16 VL elettrolitico
C3: 100 nF multistrato
C4: 100 nF multistrato
C5: 15 pF ceramico
C6: 15 pF ceramico
U1: FT232BM
T1: BC547
LD1: led rosso
LD2: led verde
Q1: quarzo 6 MHz

Varie:

- Connettore USB/B
- Connettore RJ45
- Strip maschio 4 pin
- Circuito stampato codice S0635

SCHEMA ELETTRICO



fornisce energia al resto del circuito, cui il chip è collegato tramite seriale. Volendo, è anche possibile alimentare il chip completamente dall'esterno:

questo caso la linea +5V della porta USB non viene collegata. All'interno dell'FT232BM è presente un regolatore a 3,3 V che prende tensione dal ramo positivo della sezione USB (pin 3 e 26). La tensione d'uscita, presente sul piedino 6, è ulteriormente filtrata dal condensatore

C4, che, secondo il costruttore, deve essere tenuto vicino il più possibile ai piedini 6 e 9, 17. Nel nostro schema il regolatore non viene usato, tuttavia sappiate che il produttore l'ha previsto per alimentare (tramite il

pin 6) la linea VCCIO (cioè il positivo di alimentazione della sezione di I/O seriale) nei casi in cui l'integrato debba dialogare con apparati che operano con livelli di tipo 0/3,3 V. Tale accorgimento permette, pur alimentando il resto dell'integrato con 5 volt di inviare sul lato seriale impulsi a 3,3 V. Tornando allo schema del circuito di conversione, notate il transistor T1: è collegato alla linea RSTOUT sul piedino 5; quest'ultima è l'uscita del generatore di reset interno e segue l'eventuale ingresso RST (che nel nostro caso non viene usato ed è quindi posto fisso a 5 V). Inoltre, quando viene alimentato il



Il protocollo di comunicazione tra modulo RTC e mondo esterno

Funzione	Comando	Parametro	Risposta
Imposta i secondi	*W0xx	xx indica i secondi	> xx <
Imposta i minuti	*W1xx	xx indica i minuti	> xx <
Imposta le ore	*W2xx	xx indica le ore	> xx <
Imposta giorno settimana	*W3xx	xx: 01 ÷ 07 (01: lun, 02: mar...)	> xx <
Imposta giorno mese	*W4xx	xx: 01 ÷ 31	> xx <
Imposta il mese	*W5xx	xx: 01 ÷ 12	> xx <
Imposta l'anno	*W6xx	xx: 00 ÷ 99	> xx <
Imposta la frequenza dell'onda quadra (duty 50%) disponibile sul pin SQW (S)	*W7xx	xx 00 uscita a livello logico 0 10 1Hz 11 4096 kHz 12 8192 kHz 13 32768 kHz 80 uscita a livello logico 1	> xx <
Legge i secondi	*R0	non applicabile	> 00 < ÷ > 59 <
Legge i minuti	*R1	non applicabile	> 00 < ÷ > 59 <
Legge le ore	*R2	non applicabile	> 00 < ÷ > 23 <
Legge il giorno della settimana	*R3	non applicabile	> 01 < ÷ > 07 < (01 lunedì ÷ 07 domenica)
Legge il giorno	*R4	non applicabile	> 01 < ÷ > 31 <
Legge il mese	*R5	non applicabile	> 01 < ÷ > 12 <
Legge l'anno	*R6	non applicabile	> 00 < ÷ > 99 <
Legge la frequenza dell'onda quadra (duty 50%) disponibile sul pin SQW (S)	*R7	non applicabile	> 00 < uscita a livello logico 0 > 10 < 1Hz > 11 < 4096 kHz > 12 < 8192 kHz > 13 < 32768 kHz > 80 < uscita a livello logico 1
Legge l'ora completa	*RT	non applicabile	> HH:MM:SS < > 12:24:26 <
Legge la data completa	*RD	non applicabile	> YY/MM/GG-DD < > 07/03/28-03 <
Richiede l'invio continuo della data e ora (per interrompere l'invio bisogna rinviare lo stesso comando)	*RC	non applicabile	> YY/MM/GG-DD HH:MM:SS < > 07/03/28-03 12:24:26 <

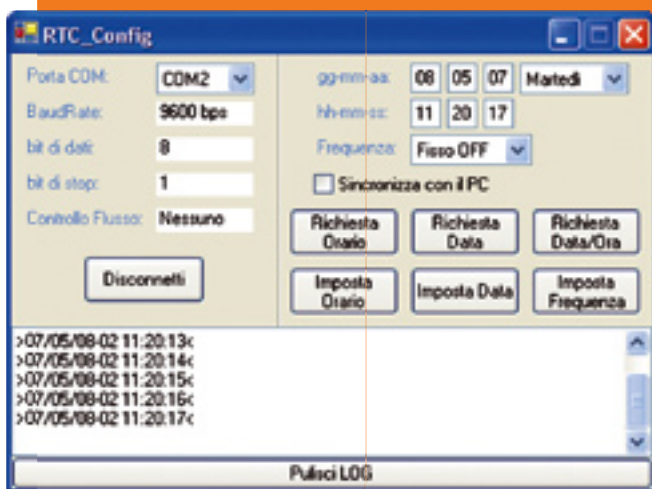
circuito, essa rimane in stato di alta impedenza fin quando la tensione di alimentazione VCC non sale sopra i 3,5 V e per ulteriori 2 millisecondi, quindi si pone a +3,3 V (tensione ricavata dal solito regolatore interno). Scopo del transistor è tenere fissa a livello alto la linea D+ dell'USB finché l'alimentazione non si stabilizza. Il convertitore FT232BM funziona con un clock di 6 MHz, ricavato mediante il quarzo Q1, che, aiutato da C5 e C6, stabilizza il funzionamento dell'oscillatore interno.

La costruzione

Il modulo RTC si realizza su una

minuscola basetta ramata le cui piste sono disegnate per componenti a montaggio superficiale; chi non se la sentisse di fare da sé, potrà acquistarlo montato e con il micro già programmato. Una volta inciso il piccolo stampato, bisogna disporvi i pochi componenti occorrenti iniziando con il microcontrollore e il DS1307: ciascuno di essi va prima appoggiato centrandone i piedini nelle rispettive piazzole, quindi fermato stagnando uno di essi e poi, uno alla volta, gli altri. Per le saldature bisogna usare un saldatore a punta molto sottile, da 25 o 30 W di potenza e del filo di lega saldan-

te avente diametro di 0,5 mm. Non dimenticatevi inoltre del flussante, indispensabile quando si effettuano montaggi con componenti SMD. Sistemato l'RTC, si deve provvedere alla basetta di interfaccia, che è molto semplice e, volendo, la si potrebbe realizzare con un pezzetto di millefori; in tutti i casi, bisogna montare su di essa il led (prestando attenzione al verso indicato nell'apposito disegno) e il connettore RJ-45. Per le connessioni con il modulo, raccomandiamo delle strip femmina a passo 2,54 mm. Infine, chi necessitasse dell'interfaccia USB potrà prepararla a partire dal circuito ➤



Configuratore RTC

Tramite l'interfaccia USB, è possibile configurare i parametri di funzionamento direttamente da un Personal Computer; allo scopo abbiamo preparato un software che gira in ambiente Windows XP, 2000, Me, facilmente installabile e utilizzabile da chiunque. Una volta avviato, il programma presenta una finestra di dialogo, che riepiloga i parametri modificabili dall'utente, suddivisa in tre parti: quella in alto a sinistra riguarda la connessione con il PC, quella alla sua destra raggruppa le impostazioni dell'RTC e la sezione in basso mostra gli eventi acquisiti, ossia le stringhe con data e ora che arrivano progressivamente. Per quanto riguarda la comunicazione, nell'apposito riquadro troviamo la casella *Porta COM*, facendo clic nella quale si apre un menu a tendina da cui selezionare la COM virtuale assegnata all'USB; in *BaudRate* si sceglie la

velocità di comunicazione e nelle tre caselle sottostanti si definiscono bit di dati, bit di stop e l'eventuale applicazione del controllo di flusso. Il pulsante in basso permette di instaurare o interrompere la comunicazione con l'interfaccia e quindi l'RTC: quando ancora non si è connessi, si chiama *Connetti*, mentre se l'interfaccia è connessa prende il nome di *Disconnetti*. Nella sezione in alto a destra troviamo le caselle con le quali si impostano ora e data attuali; quella siglata *Frequenza* dà accesso ad un menu a tendina dal quale si può definire la frequenza del clock ausiliario prelevato dall'RTC (l'impostazione predefinita è *OFF*, che corrisponde a disattivare l'uscita corrispondente). Spuntando la casella *Sincronizza con il PC*, si fa in modo che l'RTC venga aggiornato con l'ora e la data del computer. I pulsanti di richiesta (orario, data ecc.) servono ad interrogare il modulo; quelli di impostazione producono, invece, l'aggiornamento dell'RTC con i valori scritti nelle caselle in alto (*gg:mm:aa* e *hh:mm:ss*). Particolare attenzione merita *Imposta frequenza*: facendovi clic si definisce la cadenza con la quale, in modo automatico, l'RTC invia all'esterno ora e data.

stampato, da ricavare per fotoincisione seguendo l'apposita traccia. A tal proposito, vi ricordiamo che i master di tutti i circuiti presentati in questo articolo possono essere scaricati dal sito Internet della nostra rivista (www.elettronica.in.it).

Inciso e forato lo stampato dell'interfaccia, vi si possono montare i componenti iniziando con le resistenze, i diodi al silicio (attenzione alla loro polarità) e il converter seriale/USB, che deve essere del tipo per PLCC; notate che in questo caso, eccezion fatta per l'FT232BM,

i componenti sono tutti di tipo tradizionale. Sistemate poi i condensatori, dando la precedenza a quelli non polarizzati (prestare attenzione alla polarità degli elettrolitici) e i due led, da orientare come indicato, quindi il quarzo e il transistor T1. Infine, per le connessioni con il modulo RTC disponete sulla basetta una presa RJ45, da introdurre a fondo nei fori corrispondenti prima di stagnare i piedini (tutti, così da ottenere una certa robustezza); per l'USB, è stato previsto un connettore femmina di tipo "B" da c.s., che dovete inserire

a fondo e stagnare, in special modo nelle linguette di fissaggio, così da evitare il distacco accidentale nel caso, durante l'uso, il cavo venga tirato energicamente.

Per il collegamento al PC, occorre usare un cavo A/B del tipo utilizzato per stampanti e scanner. Per la connessione con il programmatore allo scopo di effettuare la programmazione in-circuit del modulo RTC (non del micro dell'interfaccia USB...) disponete sulla basetta uno strip a passo 2,54 mm di tipo maschio a quattro punte.

per il

MATERIALE

Il progetto del modulo RTC descritto in queste pagine è disponibile già montato e collaudato (cod. FT674M) al costo di 18,00 Euro. Il master ed il firmware di questo progetto, così come quelli della scheda di adattamento, possono essere scaricati gratuitamente dal sito della rivista (sito www.elettronica.in.it). Anche l'apposito software di configurazione del modulo RTC è scaricabile gratuitamente.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-778112 ~ <http://www.futuranet.it>

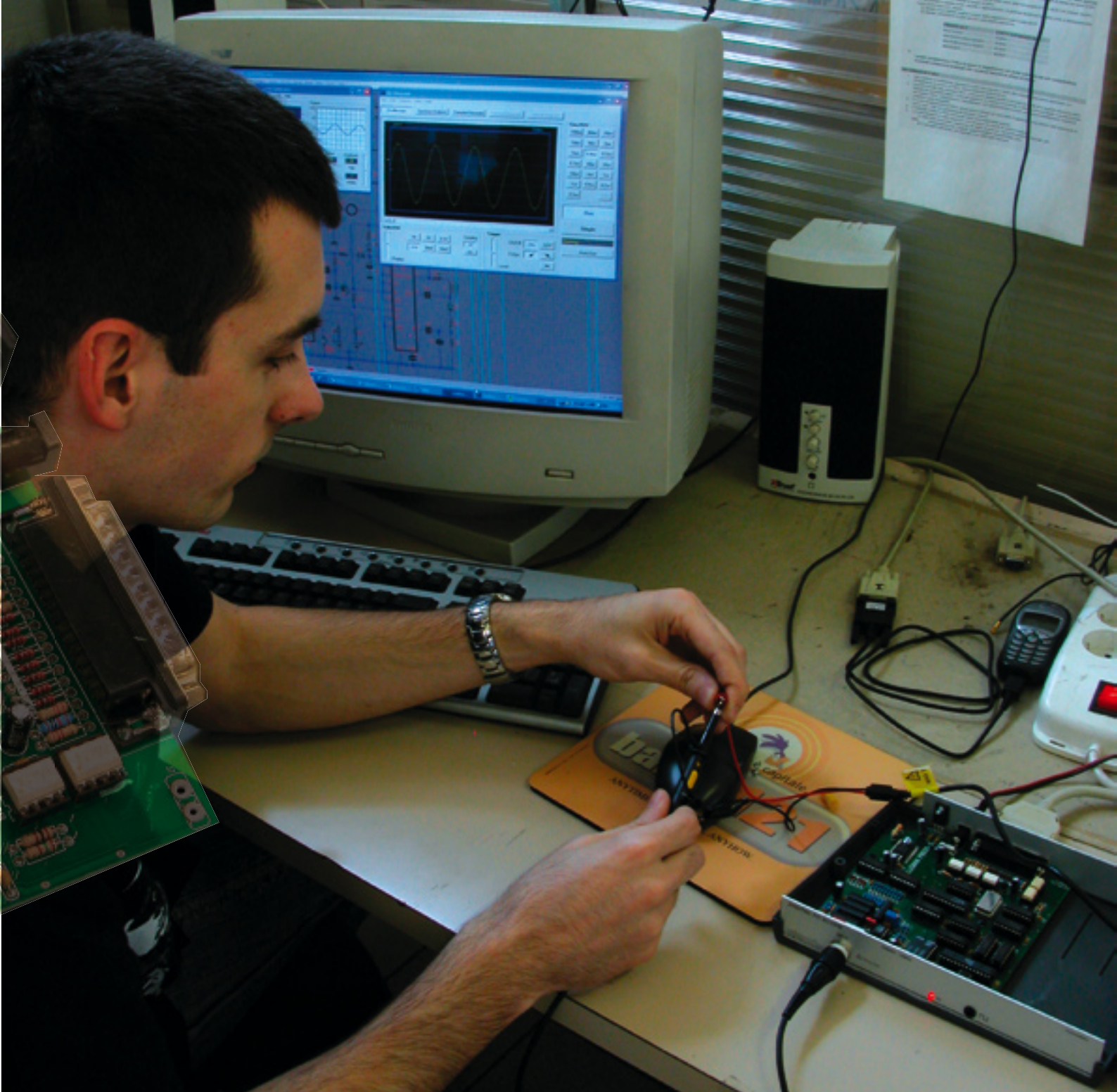
Il computer diventa un oscilloscopio

di Davide Scullino

Perché spendere soldi e occupare spazio sul banco di lavoro, quando il PC può funzionare come un perfetto strumento di misura? Basta realizzare l'interfaccia di acquisizione descritta in queste pagine e collegarla alla porta parallela, per avere un ottimo oscilloscopio da 12 MHz ed un analizzatore di spettro. Prima puntata.

La strumentazione elettronica da laboratorio, così come altri tipi di strumentazione, ha conosciuto un forte sviluppo per effetto della diffusione dei Personal Computer e del loro utilizzo non solo come dispositivi di elaborazione di dati inseriti da tastiera ma anche come sistemi in grado di elaborare dati e informazioni acquisite mediante specifiche interfacce. Sono nate così schede di acquisizione che, corredate da appositi programmi, permettono di visualizzare - sul monitor - forme d'onda ancora più accurate di quelle dei migliori oscilloscopi. Ma non solo: anche validi generatori di forme d'onda

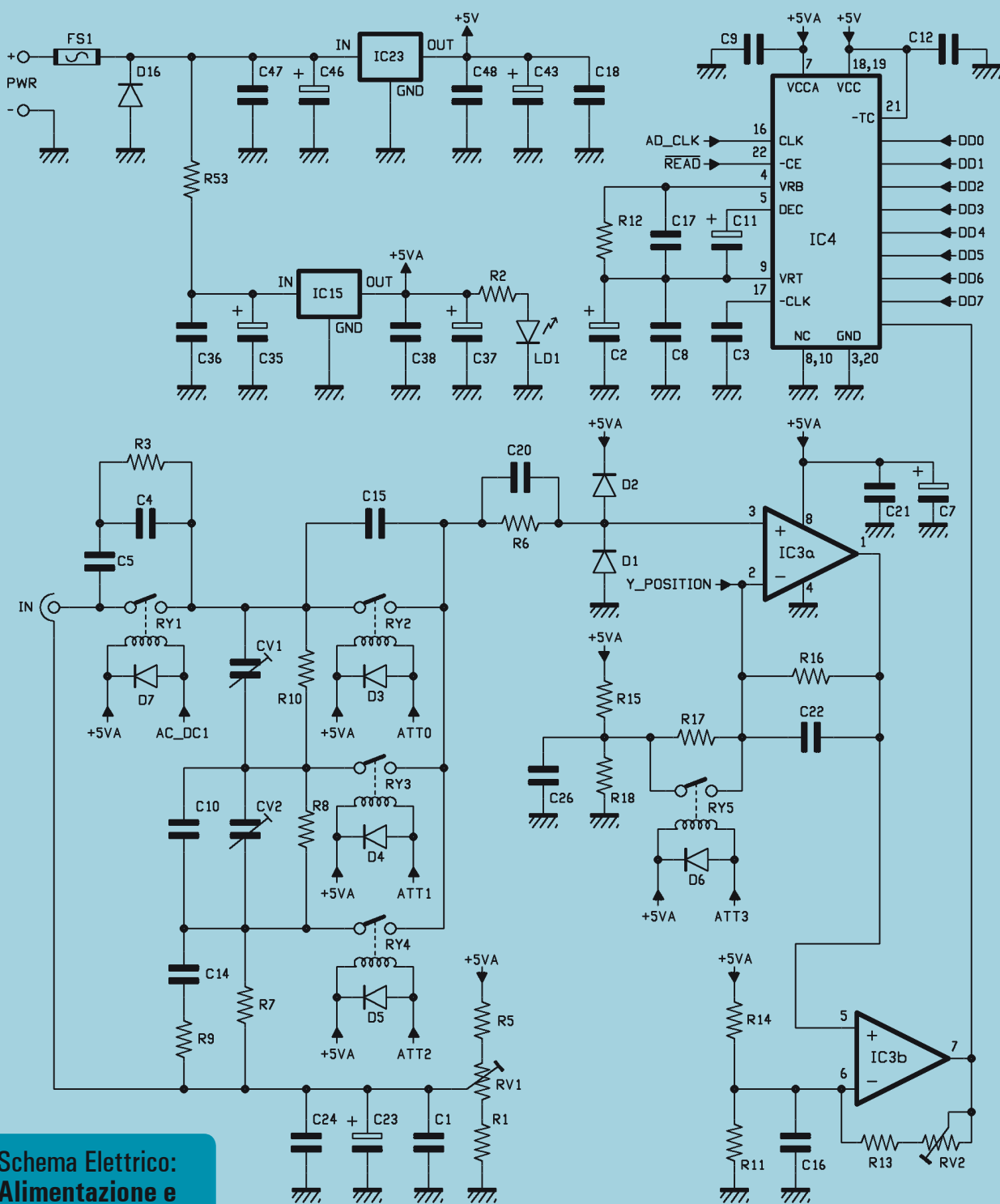
e di impulsi, analizzatori di spettro, registratori di transienti, ecc. Il vantaggio di affidare al PC le funzioni degli strumenti da laboratorio si comprende considerando le potenzialità delle CPU e l'ampia disponibilità di memoria, che permettono di decentrare le funzioni facendo svolgere quelle più complesse al computer e lasciando a circuiti molto semplici ed economici l'acquisizione dei dati. Per esempio, si pensi ai generatori di funzioni: la sintesi di segnali semplici quali l'onda triangolare, la sinusoidale e la quadra possono essere realizzate con circuiti integrati specifici, ma tensioni dall'andamento



discontinuo o particolarmente complesso richiederebbero l'impiego di moltissimi componenti; affidando il compito ad un computer, basta realizzare un software che costruisca la forma d'onda desiderata e piloti un'interfaccia esterna (ricorrendo ad una delle porte di comunicazione) estremamente semplificata, la quale non deve fare altro che convertire il dato da digitale ad analogico. Anco-

ra, si immagini il beneficio nell'analisi delle forme d'onda: memorizzare l'andamento di una tensione per vederne un dettaglio richiede, con l'elettronica tradizionale, soluzioni che vanno dalla persistenza introdotta negli oscilloscopi Hewlett Packard al campionamento ed al salvataggio in una memoria RAM negli strumenti digitali. L'inevitabile complessità circuitale richiesta

da tali soluzioni può essere aggirata ancora con l'aiuto del PC: l'acquisizione di un certo periodo di un segnale al fine di rivederlo in tutto o in parte per studiare quel che un normale oscilloscopio non permetterebbe di vedere, la comparazione di due segnali o alcune analisi di spettro, la verifica delle distorsioni, possono essere realizzate con una semplicissima interfaccia in grado di inviare ➤



**Schema Elettrico:
Alimentazione e
ingresso sonda**

al computer, in formato digitale, la tensione letta. Il software provvede poi ad elaborare i dati nel modo più appropriato. Sono proprio la semplicità ed il basso costo, a rendere appetibile la strumentazione da PC; ed è perciò che abbiamo pensato di

dedicare un po' di pagine all'argomento, proponendo la realizzazione di un'interfaccia da collegare sulla porta parallela (quella normalmente destinata al collegamento della stampante) che permetta di acquisire segnali elettrici, digitalizzarli e

passarli al computer, nel quale un apposito programma (scaricabile gratuitamente dal Web e personalizzabile grazie alla disponibilità di un'apposita DLL) permetterà di realizzare non solo la funzione di visualizzazione, ma anche quella

di registrazione dei transienti (memoria) e analisi di spettro. Quindi, tre strumenti in uno: oscilloscopio, analizzatore di spettro e registratore di eventi.

Lo strumento è a singola traccia ed ha un'impedenza d'ingresso di 1 Mohm su 30 pF di capacità; la banda passante (± 3 dB) è compresa tra 0 Hz e 12 MHz, mentre la massima tensione sopportabile è 100 V in continua o alternata (si parla, in questo caso, del valore efficace) e l'accoppiamento, impostabile da computer, può essere in AC o in DC.

La sezione di comunicazione dell'oscilloscopio con la porta parallela è optoisolata, così da evitare interferenze o danni accidentali; la separazione galvanica permette di mantenere isolata la massa del computer da quella dell'interfaccia di acquisizione e quindi del circuito in esame. In tal modo è possibile compiere misure anche su dispositivi collegati alle altre porte del PC, ovvero sulla rete, senza rischiare che, una differenza tra i potenziali di terra (dovuta, ad esempio, ad un cattivo isolamento nell'alimentatore o nel monitor) possa scaricare corrente dalla parallela al nostro strumento o viceversa, determinando danni non indifferenti.

Schema elettrico

L'interfaccia di misura è decisamente complessa, quindi per analizzarla proviamo a scomporla nelle parti essenziali da cui è formata, che sono: la sezione di ingresso, la parte digitale e la sezione optoisolata di comunicazione con il computer. Partiamo dando uno sguardo alla prima, che inizia dal connettore BNC col quale lo strumento preleva dalla sonda il segnale da analizzare; come potete notare, la presa non è riferita alla massa di alimentazione ma ad uno "zero artificiale" il cui potenziale viene definito accuratamente con il trimmer multigiri RV1. Ciò vale in corrente continua, perché nelle mi-

L'HARDWARE

Sebbene nasca per trasformare in oscilloscopio un Personal Computer, il dispositivo descritto in queste pagine permette di realizzare più funzioni, rese possibili dal fatto che, una volta acquisito un segnale, lo si può analizzare, scomporre e memorizzare per poi rivederlo anche a singoli frammenti. Le caratteristiche più significative del dispositivo sono:

- impedenza d'ingresso: 1 Mohm/30 pF
- risposta in frequenza (-3 dB): 0 Hz ÷ 12 MHz
- risoluzione verticale (in ampiezza): 8 bit
- impostazione automatica della base dei tempi e dei V/div.
- funzione memoria

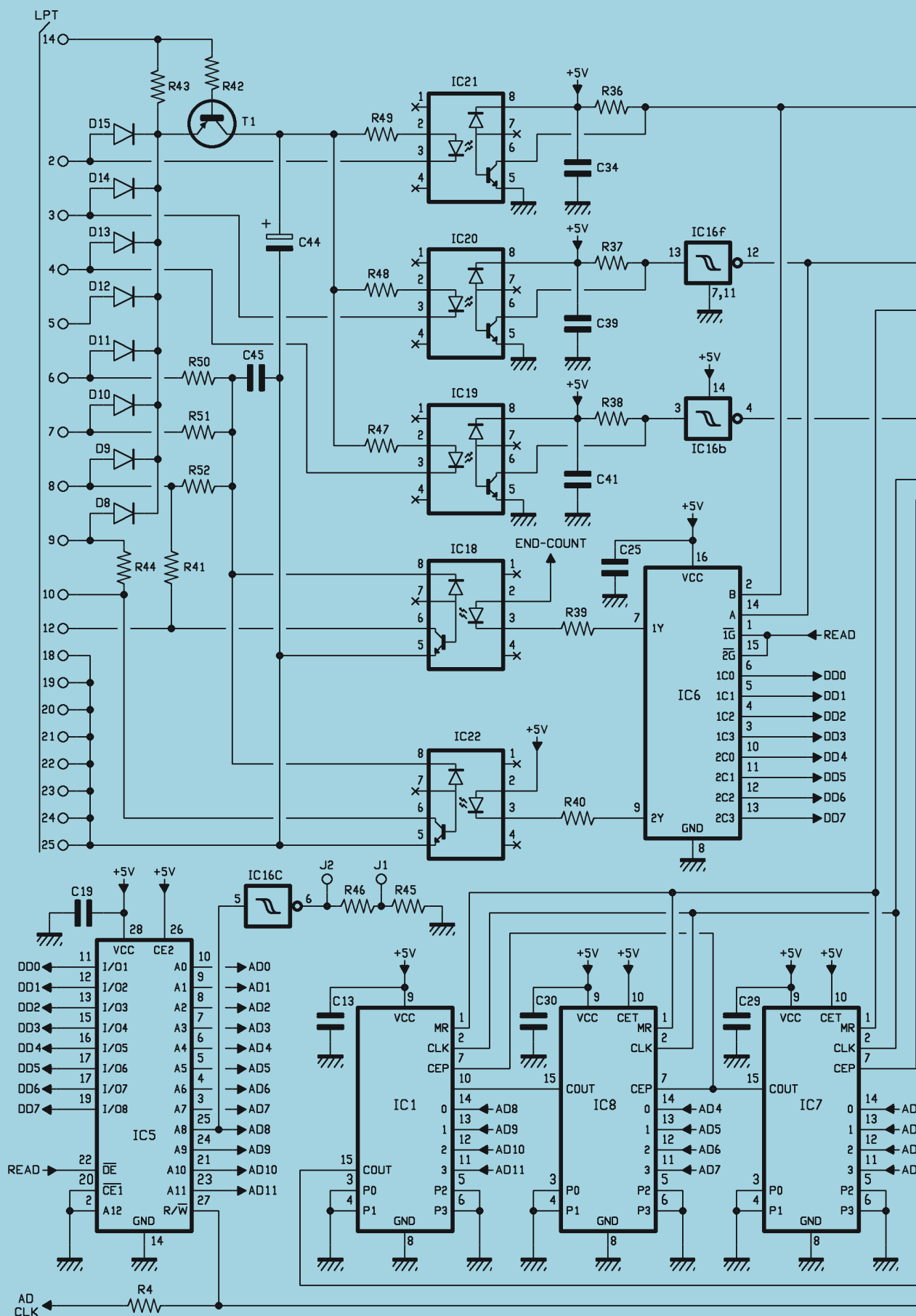
L'OSCILLOSCOPIO

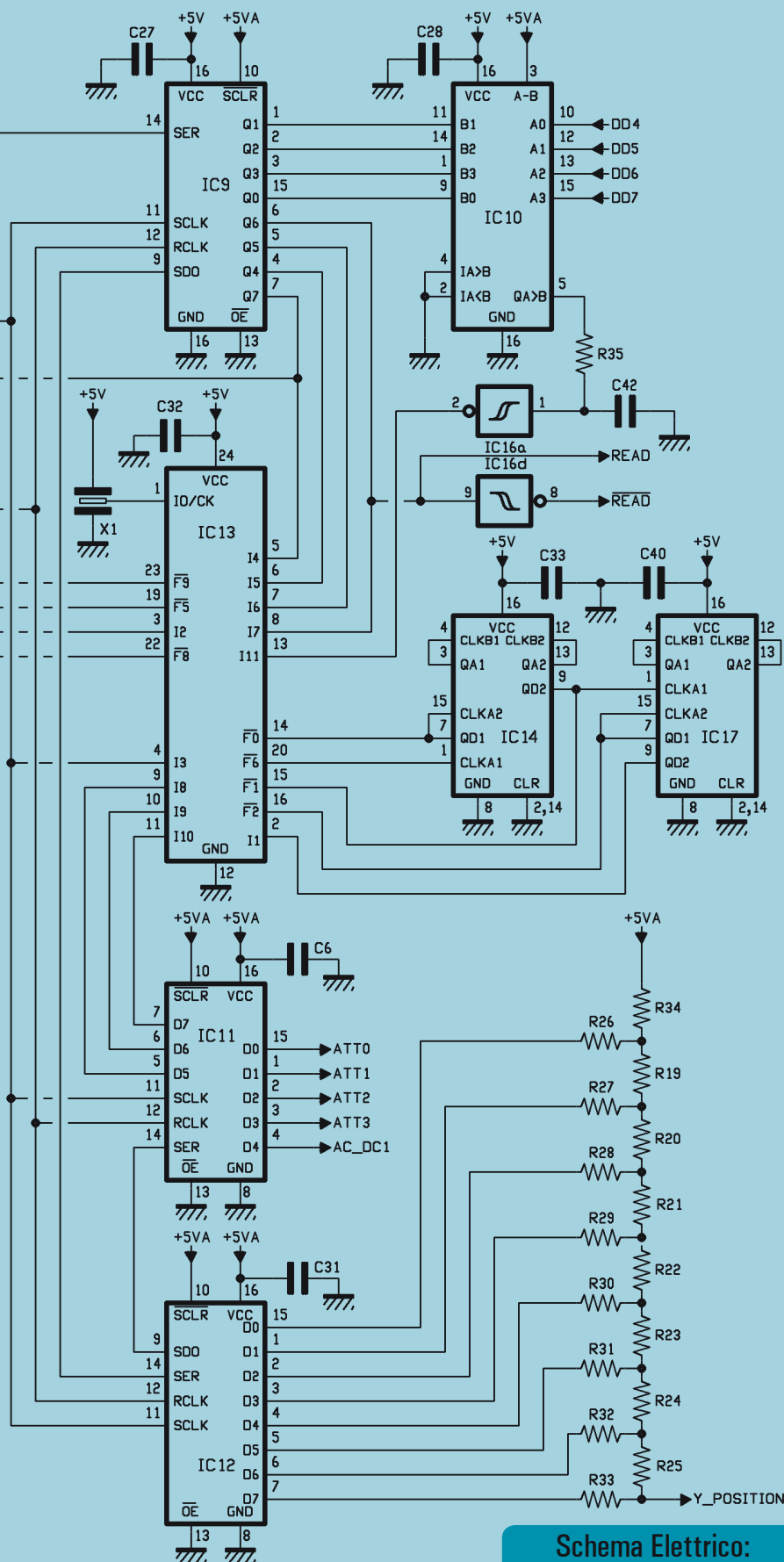
Nella funzione oscilloscopio, lo strumento di misura permette di visualizzare anche in modo differito qualsiasi forma d'onda di frequenza compresa fra 0 e 12 MHz e ampiezza non eccedente i 100 V (100 V_{eff}. In corrente alternata). Le caratteristiche principali sono:

- singola traccia
- base dei tempi: 0,1 μ s ÷ 100 ms /div.
- sorgente di trigger: su canale 1 o indipendente
- livello di trigger: regolabile ogni ½ divisione
- sensibilità d'ingresso: 10 mV ÷ 3 mV/div.
- dimensione record: 4079 campionamenti
- frequenza di campionamento: 800 Hz ÷ 32 MHz
- lettura immediata del valore RMS per le misure AC
- marcatori per frequenza, tempo e ampiezza.

sure in alternata i condensatori C1, C23 e C24 bypassano il trimmer ed R1. Il conduttore "caldo" passa da una rete resisto-capacitiva che permette di determinare l'attenuazione d'ingresso e il tipo di accoppiamento, in base alle impostazioni date dal computer; per l'esattezza, il relé RY1 quando è eccitato chiude in cortocircuito il bipolo R3/C4 ed il C5 realizzando l'accoppiamento in continua, mentre stando a riposo obbliga il segnale a transitare da tale rete e quindi fa sì che l'oscilloscopio sia accoppiato in alternata al punto di misura. I tre relé RY2, RY3, RY4 e le resistenze collegate ai loro piedini 1 servono a decidere il livello di attenuazione del segna-

le, quindi la sensibilità dell'ingresso dello strumento; quando è eccitato il solo RY2, il suo scambio trasporta la componente da analizzare direttamente all'ingresso della rete di amplificazione, mentre se tale relé risulta a riposo ed è eccitato il solo RY3, si introduce una consistente attenuazione. Infatti, in tal caso, il segnale deve attraversare la resistenza R10, che fa partitore con R6 ed R7; essendo la somma di queste ultime pari a circa 100 kohm ed R10 da 910 kohm, il risultato è un rapporto di partizione di 1:10, quindi il segnale subisce un'attenuazione pari a 10 volte (-20 dB). Infine, se è il solo RY4 ad essere eccitato, il suo scambio costringe il segnale ➤





d'ingresso ad attraversare sia R10 che R6, quindi stavolta il partitore responsabile dell'attenuazione è formato dalla somma di tali resistori e da R7. Dai valori di tali componenti si deduce che la componente d'ingresso subisce un'attenuazione di 100 volte (-40 dB).

Il blocco di attenuazione permette di mantenere costante il campo di ampiezza della tensione inviata al convertitore analogico/digitale, così da semplificare poi la lettura e l'interpretazione dei dati corrispondenti da parte del software di gestione.

Dopo il selettore di attenuazione, il segnale passa al blocco formato dagli operazionali IC3a e IC3b, scopo del quale è definire la posizione verticale della traccia; in altre parole, in esso viene sommata una certa componente continua che determina lo spostamento dell'ampiezza rispetto allo zero di riferimento.

Il potenziale di shift viene determinato dal software di controllo, intervenendo sull'integrato IC12 (74HC595) della sezione digitale al fine di costruire la tensione desiderata mediante il partitore multiplo formato dalle resistenze R19÷R34. A seconda della combinazione logica alle uscite del 74HC595, sul piedino 2 dell'IC3a risulta una tensione differente; essendo otto le linee dell'IC12 e potendo assumere ciascuna due stati logici (0 e 5 V) si può affermare che l'accuratezza dell'impostazione è 8 bit, ovvero sono possibili 256 livelli di potenziale.

Considerato che lo stadio d'ingresso è stato calcolato per lavorare con tensioni dell'ordine del volt (1 Veff. in caso di misura in alternata) e che quindi l'uscita dell'operazionale IC3a può registrare una variazione di ± 2 V rispetto allo zero di riferimento, la tensione riportata all'ingresso invertente non può superare 3 V né abbassarsi sotto i 2 V; ciò significa che è prevista un'escursione di 1 V.

Essendo possibili 256 livelli di ten- ➤

sione, l'accuratezza dello spostamento è di circa 10 millivolt per livello: decisamente più di quello che si può ottenere ruotando manualmente una tradizionale manopola.

Notate che nello stadio di spostamento verticale è inserito anche il relé RY5, che interviene sul guadagno dell'amplificatore IC3a: quando è eccitato, il suo scambio cortocircuita R17 e determina un aumento del guadagno, quindi della sensibilità dell'ingresso; se, invece, RY5 si trova a riposo, l'amplificazione dell'operazionale è minore e si abbassa perciò la sensibilità d'ingresso dello strumento, ovvero viene introdotta un'ulteriore attenuazione da sommarci a quella eventualmente operata da RY2, RY3 ed RY4.

Dal piedino 7 dell'IC3b il segnale amplificato ed eventualmente addizionato della componente continua di spostamento verticale, viene applicato all'ingresso del convertitore A/D siglato IC4; si tratta di un TDA8703 che lo converte, con una risoluzione di 8 bit (a tanto ammonita la risoluzione verticale dell'oscilloscopio) in formato binario e passa i rispettivi dati al bus di comunicazione con l'unità digitale. Vedremo tra breve che fine fanno i dati.

L'acquisizione viene effettuata ad una frequenza che può variare tra 800 Hz e 32 MHz, in base alle impostazioni date dall'operatore nella finestra di dialogo del programma di gestione da PC, ovvero a seconda della frequenza del segnale d'ingresso; il campionamento si effettua quando l'A/D converter viene abilitato mediante l'applicazione dello zero logico sulla propria linea /CE (piedino 22) da parte del terminale /READ in arrivo dalla solita unità digitale, che fornisce anche il clock (sulla linea ADCLK) da cui dipende la frequenza del campionamento.

L'intero oscilloscopio funziona a tensione continua, applicata al plug di alimentazione, di valore compreso fra 9 e 15 V; assorbe una corren-

IL REGISTRATORE DI TRANSIENTI

Oltre che da oscilloscopio, l'interfaccia può essere usata come registratore di eventi, ossia di variazioni nella tensione letta; a ciò provvede il software in dotazione, che, una volta acquisito il segnale, permette di decidere quale intervallo memorizzare e quali eventi salvare e rivedere sullo schermo. La funzione è molto utile per vedere cosa accade in circuiti soggetti a disturbi casuali e nei dispositivi a commutazione.

Le caratteristiche del registratore di eventi sono:

- scala dei tempi: 20 ms ÷ 2000 s/div.
- massima durata di registrazione: 9,4 ore per schermata
- memorizzazione automatica dei dati
- registrazione automatica per oltre 1 anno
- marcatori per lettura di tempo e ampiezza
- funzione zoom

L'ANALIZZATORE DI SPETTRO

Completa il set di funzioni realizzabili con l'interfaccia descritta in queste pagine ed è estremamente utile per tracciare i grafici di risposta in frequenza e banda passante di filtri, amplificatori, diffusori acustici, apparecchi per la riproduzione del suono. Le caratteristiche della funzione sono:

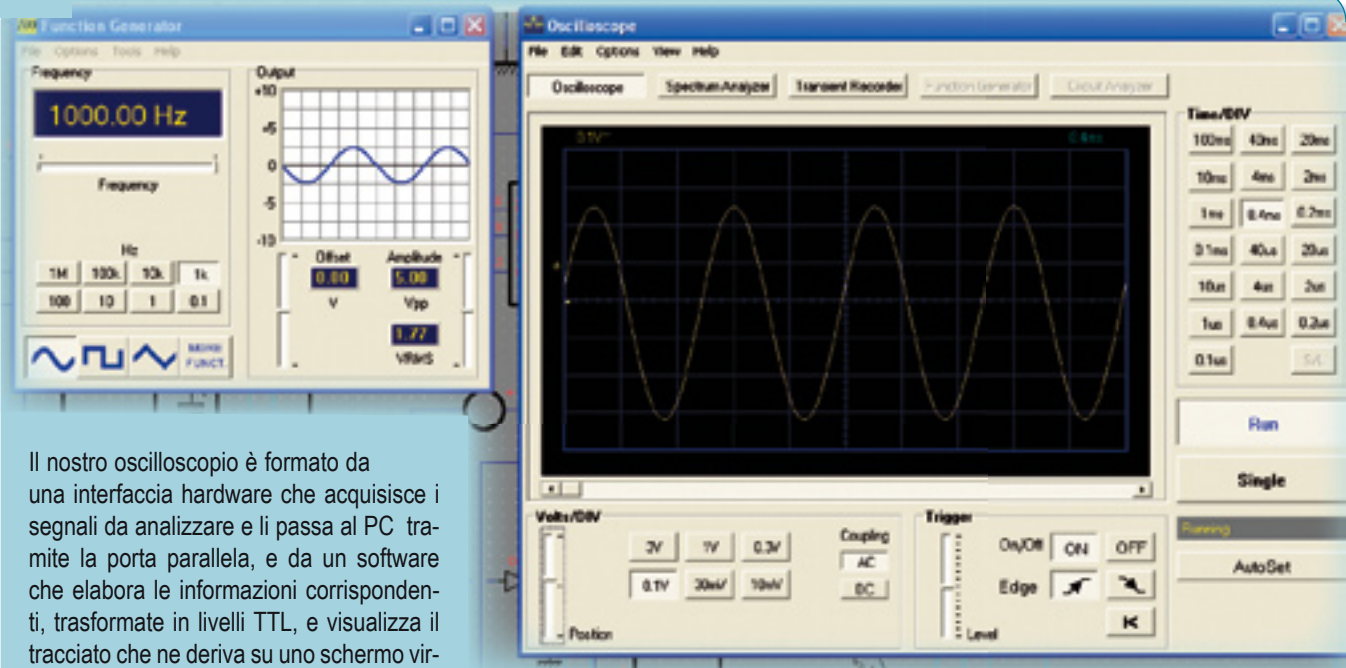
- campo di misura: 0 Hz ÷ 16 MHz
- scala dei tempi lineare o logaritmica
- funzione zoom

te che può raggiungere 500 mA. La componente continua passa dal fusibile di protezione e raggiunge gli ingressi di due regolatori 7805, ciascuno dei quali fornisce 5 V ben stabilizzati; per evitare interferenze fra la sezione digitale e quella analogica (ingresso del segnale) si è preferito distinguere le alimentazioni: IC23 alimenta la logica (+5 V) e i piedini Vccd dell'A/D converter (sezione digitale del TDA8703) mentre IC24 serve a far funzionare gli stadi d'ingresso, di spostamento verticale, i relé e la parte analogica dell'A/D converter (Vcca).

Osservate il diodo di protezione posto in parallelo all'ingresso di alimentazione: se si applica tensione con la polarità opposta, va in conduzione e provoca un assorbimento tale da far bruciare il fusibi-

le, evitando così danni al resto del circuito; normalmente deve essere interdetto.

Passiamo adesso alla parte digitale, che è poi quella più sostanziosa: in essa si trovano una PAL, un multiplexer, quattro shift register a 8 bit di tipo 74HC595, una RAM statica 6264 (da 8 k x 8 bit) tre contatori sincroni 74F161 ed altri quattro contatori ripple-carry binari/decimali 74HC390, più qualche porta logica NOT. L'unità nell'insieme serve a dialogare con il computer, avvalendosi dell'interfaccia optoisolata, al fine di eseguire tutte le operazioni richieste dall'operatore tramite il programma di gestione. Più esattamente, comanda l'A/D converter per effettuare il campionamento del segnale d'ingresso eventualmente traslato in



Il nostro oscilloscopio è formato da una interfaccia hardware che acquisisce i segnali da analizzare e li passa al PC tramite la porta parallela, e da un software che elabora le informazioni corrispondenti, trasformate in livelli TTL, e visualizza il tracciato che ne deriva su uno schermo virtuale. È noto che l'oscilloscopio può essere usato per moltissime misure su segnali elettrici, ma spesso coopera con un generatore di forme d'onda, ad esempio quando si deve analizzare il comportamento di un filtro o di un amplificatore nei confronti della frequenza. Il generatore di forme d'onda abbinabile all'oscilloscopio è il PCG10, già descritto nel fascicolo n°68: anch'esso si connette al computer tramite la porta parallela. A chi si domanda come usare contemporaneamente oscilloscopio e generatore di segnali con un PC che generalmente dispone solo di una porta parallela, rispondiamo che la cosa è possibile: la porta viene collegata, nel nostro caso, al generatore che la ripete

su un secondo connettore, così da poter connettere in cascata l'interfaccia dell'oscilloscopio. Il programma che gestisce gli strumenti provvede poi ad istruire la parallela in modo che venga considerato un solo dispositivo alla volta, così da evitare un conflitto di risorse. In pratica, quando il computer deve comunicare con l'oscilloscopio inibisce il buffer del generatore e viceversa. Grazie a questo accorgimento, il PC può diventare al tempo stesso generatore di segnali (in alto a sinistra potete vedere l'interfaccia grafica), oscilloscopio (in alto a destra), analizzatore di spettro e registratore di transienti.

base alle impostazioni volute, stiva i dati corrispondenti nella RAM statica e li va poi a leggere quando desiderato, ovvero in real-time se non è richiesta alcuna funzione di memorizzazione.

Il campionamento lo scandisce la PAL, inviando (tramite AD_CLK) il clock richiesto dal PC; la frequenza corrispondente dipende dall'impostazione dalle linee C0, C1, C2, che definiscono l'eventuale fattore di divisione del valore di base generato dall'oscillatore di clock X1 (32,00 MHz). Più precisamente, nella determinazione della frequenza generata sono coinvolti i contatori IC14a, IC14b, IC17a e IC17b, che, collegati in cascata, permettono di dividere fino ad un massimo di 65536 volte; la PAL emette, dalla linea F_OUT, il clock originario

ricavato dall'oscillatore, che passa al primo 74HC390. Ora, siccome di quest'ultimo si preleva il segnale dall'uscita di peso 2^4 , ciò che ne risulta è un clock diviso per 16, che si rende disponibile sulla linea F1, e raggiunge sia l'F1 della PAL, sia l'ingresso di clock dell'IC14b; in quest'ultimo avviene una seconda divisione per 16, quindi ad F2 e all'input dell'IC17a giunge un clock diviso 256 volte. IC17a divide ancora per 16, restituendo ad F3 della PAL ed al pin 15 dell'IC17b un clock diviso, complessivamente, per 4096. Infine, dall'uscita QD, il clock diviso per 65536 si presenta alla linea F4 della PAL. Quest'ultima, in base alle impostazioni date dal computer, provvederà a scegliere, fra le frequenze disponibili su F1, F2, F3, F4, il clock da inviare all'A/D

converter e quindi la frequenza del campionamento. L'attivazione del convertitore e quindi l'acquisizione della lettura viene controllata dal PC tramite le linee RCK e CLK1 ricavate, dalla porta parallela (rispettivamente dai pin 3 e 4) mediante l'interfaccia optoisolata; la prima scandisce il clock per i registri a scorrimento 74HC595 e la seconda è utilizzata come canale seriale dei dati dal PC agli stessi shift-register, che, nel circuito, servono ad impostare i singoli blocchi secondo le istruzioni impartite dal computer. Nel caso dell'A/D converter, il registro IC9 passa all'inverter logico 74HC14 il criterio che attiva il campionamento mediante l'applicazione dello stato basso sulla linea /READ. L'opposto, ossia READ (non invertito) raggiunge la PAL, ➤

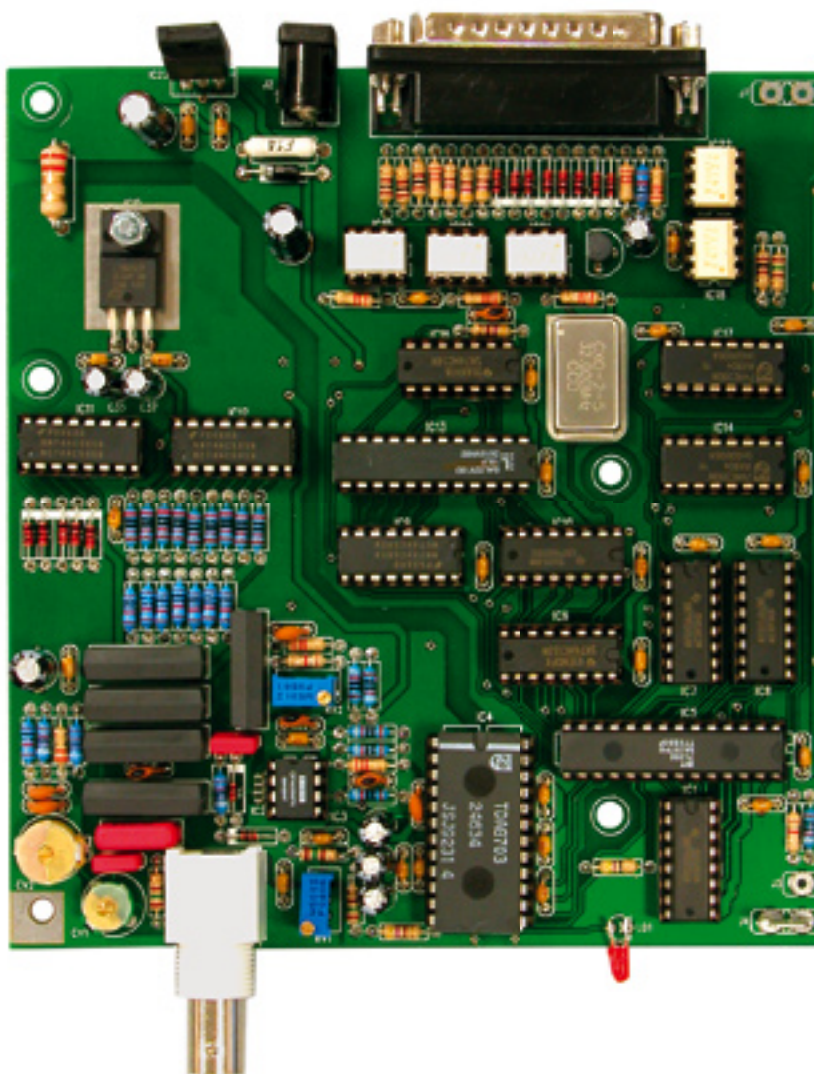
ELENCO COMPONENTI:

R1: 150 ohm
 R2: 470 ohm
 R3: 150 ohm
 R4: 47 ohm
 R5: 150 ohm
 R6: 91 kohm 1%
 R7: 10 kohm 1%
 R8: 91 kohm 1%
 R9: 3,3 ohm
 R10: 910 kohm 1%
 R11: 820 ohm 1%
 R12: 2,2 kohm
 R13+R15: 820 ohm 1%
 R16: 1,8 kohm
 R17: 2,2 kohm
 R18: 820 ohm 1%
 R19+R25: 10 kohm 1%
 R26+R34: 20 kohm 1%
 R35: 470 ohm
 R36+R38: 2,2 kohm
 R39: 100 ohm
 R40: 150 ohm
 R41, R42: 2,2 kohm
 R42: 2,2 kohm
 R43: 10 kohm 1%
 R44: 2,2 kohm
 R45: 820 ohm 1%
 R46: 1,8 kohm
 R47+R49: 100 ohm
 R50+R52: 470 ohm
 R53: 3,3 ohm 1W
 RV1: trimmer 50 ohm multigiri
 RV2: trimmer 500 ohm multigiri
 C1, C3, C6: 100 nF multistrato
 C2: 4,7 µF 50 VL elettrolitico
 C4: 2,2 nF 100 VL poliestere
 C5: 47 nF 250 VL poliestere
 C7: 4,7 µF 50 VL elettrolitico
 C8, C9: 100 nF multistrato
 C10: 220 pF ceramico
 C11: 4,7 µF 50 VL elettrolitico
 C12, C13: 100 nF multistrato
 C14: 2,7 nF multistrato
 C15, C16: 22 pF ceramico
 C17: 100 pF ceramico
 C18, C19: 100 nF multistrato

C20: 2,2 nF 100 VL poliestere
 C21: 100 nF multistrato
 C22: 22 pF ceramico
 C23: 100 µF 25 VL elettrolitico
 C24, C25: 100 nF multistrato
 C26: 100 pF ceramico
 C27+C34: 100 nF multistrato
 C35: 4,7 µF 50 VL elettrolitico
 C36: 100 nF multistrato
 C37: 4,7 µF 50 VL elettrolitico
 C38+C41: 100 nF multistrato
 C42: 22 pF ceramico

C43: 100 µF 25 VL elettrolitico
 C44: 4,7 µF 50 VL elettrolitico
 C45, C47, C48: 100 nF multistrato
 C46: 100 µF 25 VL elettrolitico
 CV1: compensatore 22 pF
 CV2: compensatore 50 pF
 D1, D2: BAS45A
 D3+D15: 1N4148
 D16: 1N4007
 T1: BC327
 IC1: 74F161
 IC3: OPA2350PA

IC4: TDA8703
 IC5: 6264
 IC6: 74HC153
 IC7, IC8: 74F161
 IC9, IC11, IC12: 74HC595
 IC10: 74HC85
 IC13: PAL22V10 (VK8031)
 IC14: 74HC390
 IC15: 7805
 IC16: 74HC14
 IC17: 74HC390
 IC18+IC22: 6N136

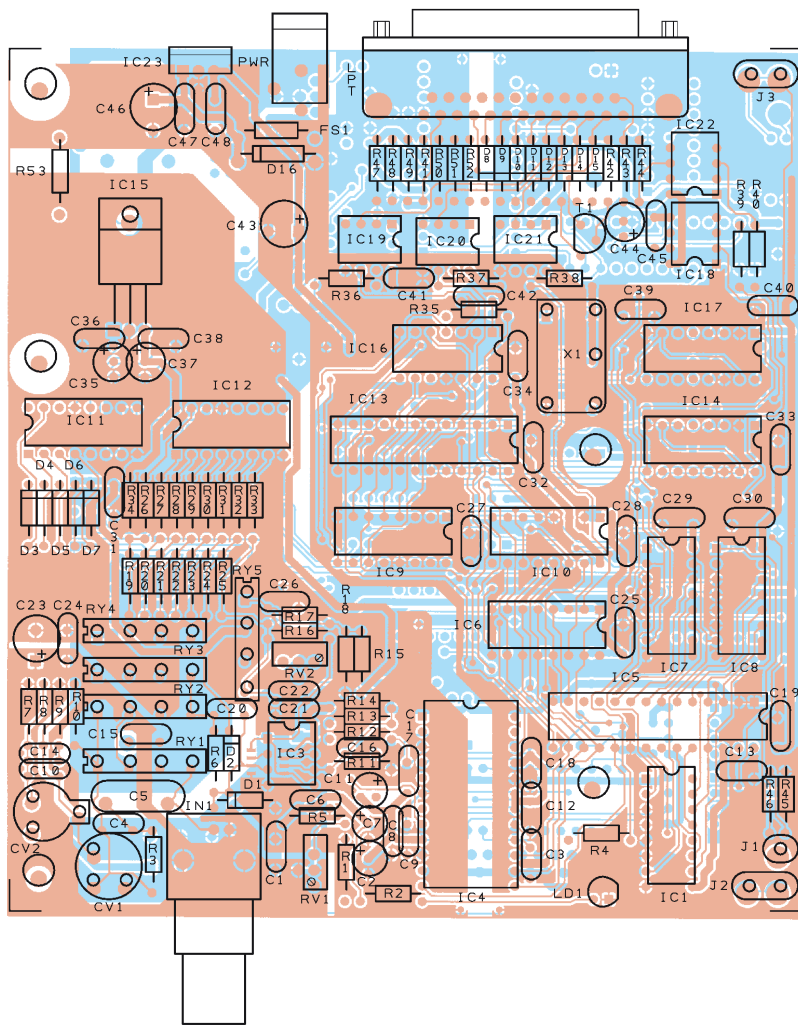


affinché essa provveda al necessario clock.

Lo shift register IC12 viene utilizzato per definire lo spostamento verticale della traccia, ovvero per costruire la corrispondente tensione impostando l'opportuna combinazione delle sue otto linee di uscita, linee che rappresentano il byte bi-

nario passato dal computer lungo la porta parallela; tutti gli 8 bit a livello logico alto determinano il massimo valore di shift, mentre con otto stati bassi la traccia viene spostata più in basso possibile, dato che la tensione corrispondente allo zero di riferimento del segnale si abbassa al minimo. Il registro IC11 si occupa,

invece, di smistare i comandi diretti ai relé che decidono i quattro possibili livelli di attenuazione (tre sull'ingresso del segnale ed il quarto sulla retroazione dell'operazionale IC3a); in virtù del ridotto assorbimento delle bobine, le sue uscite possono pilotare direttamente i relé (sono richiesti solo i diodi di pro-



Piano di montaggio dell'oscilloscopio. Le dimensioni sono state ridotte all'80%. Il master in dimensioni reali è scaricabile dal sito della rivista.

IC23: 7805

X1: Oscillatore 32MHz

RY1: Relé reed (VR05R051AS)

RY2: Relé reed (VR05R051AS)

RY3: Relé reed (VR05R051AS)

RY4: Relé reed (VR05R051AS)

RY5: Relé reed (VR05R051AS)

LD1: led 3 mm rosso

FS1: Pico fusibile 1A

Varie:

- Plug alimentazione

- Connettore BNC da cs

- Connettore DB25 maschio

- Fast-On maschio verticale da cs

- Zoccolo 4+4 (6 pz.)

- Zoccolo 7+7

- Zoccolo 8+8 (10 pz.)

- Zoccolo 12+12 (2 pz.)

- Zoccolo 14+14

- Contenitore plastico

- Vite 3 MA 10 mm (2 pz.)

- Vite 3 MA 22 mm (2 pz.)

- Dado 3 MA (2 pz.)

- Circuito stampato

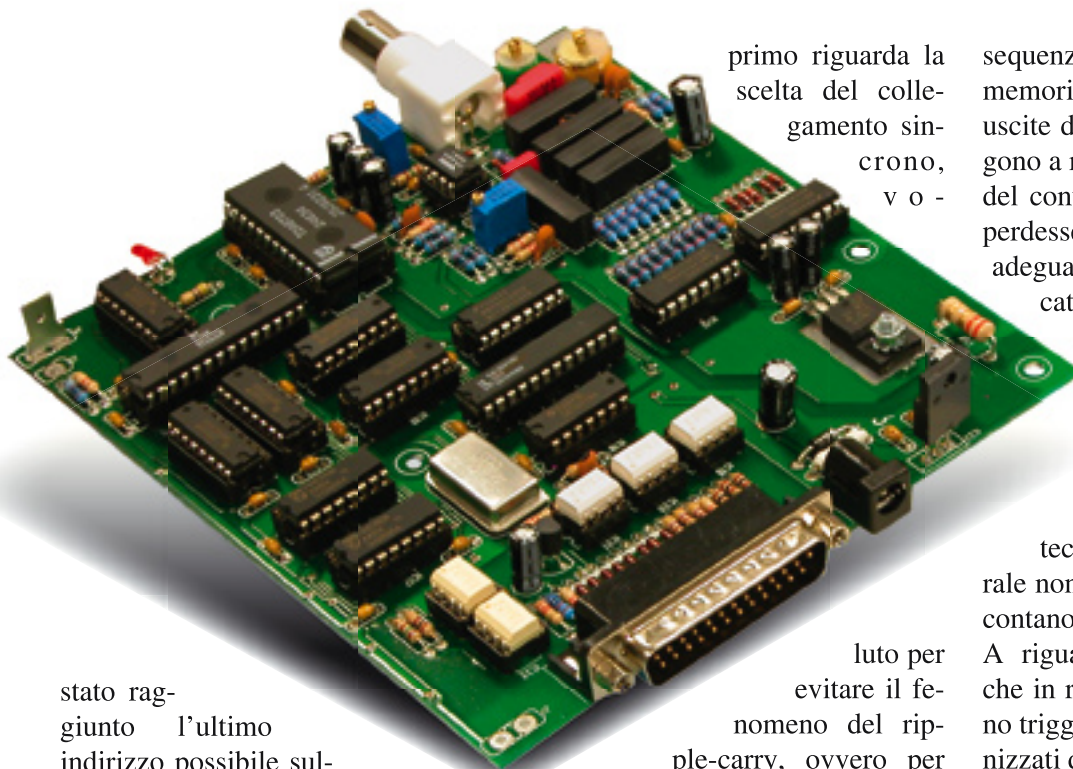
tezione D3, D4, D5, D6 e D7). Lo stesso 74HC595 comanda RY1, che decide il tipo di accoppiamento dell'input (DC o AC). Si noti che, per semplificare la struttura circuitale e risparmiare linee di collegamento al computer, i dati corrispondenti a ciò che deve uscire dai registri a scorrimento vengono inviati serialmente

su una terza linea seriale (DSR) ricavata dal piedino 2 della porta parallela e scanditi da RCK e CLK1; ogni stringa contiene tutti i dati finora elencati per IC9, IC11 e IC12, che passano da un registro all'altro entrando nel pin SER (14) e uscendo dal SDO (Serial Data Output, piedino 9). Sono proprio SER e SDO a

permettere il collegamento in cascata di più registri. Ogni 74HC595 contiene sia uno shift-register che uno storage-register; quest'ultimo serve a memorizzare i dati indirizzati all'integrato e a conservarli per poi passarli alle uscite corrispondenti. Lo storage-register viene gestito dal segnale RCK1, in modo da attivare l'aggiornamento del latch di uscita del 74HC595 solo quando contiene i dati diretti ad esso e non quelli degli altri registri componenti la serie IC9, IC11, IC12.

In ultimo, dell'unità digitale dell'oscilloscopio vediamo la sezione facente capo ai contatori sincroni, utilizzata per indirizzare la memoria da 64 kbit; le linee COUNT e TRIGGER vengono gestite direttamente dalla PAL (IC13) in modo da sincronizzare l'indirizzamento delle locazioni con il latch di uscita dell'A/D converter al fine di garantire che ogni qual volta il segnale venga campionato e i corrispondenti dati su 8 bit siano resi disponibili sul bus DD0÷DD7, la memoria possa salvarli nello stesso tempo. Dunque, mediante COUNT la PAL invia il clock ai tre 74F161, connessi secondo uno schema abbastanza tradizionale: gli ingressi CLK sono collegati tra loro in parallelo in modo da essere sincronizzati, ma i dispositivi risultano in cascata. Il clock è uguale per tutti e viene inviato continuamente; quando il primo contatore raggiunge il proprio limite massimo, ovvero quando i suoi quattro bit (AD0, AD1, AD2, AD3) hanno compiuto tutte le possibili combinazioni, attiva la propria uscita COUT (Carry Out) con la quale va a triggerare la seconda (IC8a). Sotto l'effetto del clock, il secondo inizia a contare e, raggiunto il limite, attiva anch'esso la Carry Out e con essa comanda il terzo ed ultimo contatore.

Terminato il conteggio, IC1a pone a livello alto la propria COUT e comunica alla PAL, tramite la linea E_COUNT di quest'ultima, che è ➤



stato raggiunto l'ultimo indirizzo possibile sulle linee AD0÷AD3 del primo contatore, AD4÷AD7 del secondo e AD8÷AD11 dell'ultimo; IC13 prende atto della condizione e la comunica al computer, il quale azzerla la catena dei contatori mediante la linea RESET prelevata da Q7 dello shift-register IC9. Lo stesso RESET viene inviato alla PAL, della quale raggiunge il piedino 14.

Ora la PAL sospende il clock sulla linea COUNT e la catena di contatori 74F161 rimane a riposo, almeno fin quando non siano richiesti una nuova operazione di memorizzazione o un ciclo di lettura; si noti, a riguardo, che per memorizzare la lettura dell'A/D converter nella RAM il computer invia lo zero logico sulla linea READ (/DE della memoria) e la PAL (che legge lo stesso READ tramite il proprio piedino 8) pone la linea WE (R/W, ossia pin 27) a zero logico. Invece, quando occorre leggere il contenuto della memoria, si ha ancora l'attivazione di READ, ma la PAL fornisce alla WE il livello alto: ciò determina la condizione di lettura (il piedino R/W deve stare ad 1 logico in lettura e a zero in scrittura). Della catena di contatori vanno notati alcuni ulteriori dettagli: il

primo riguarda la scelta del collegamento sincrono, v o -

luto per evitare il fenomeno del ripple-carry, ovvero per assicurarsi che la composizione dei bit di indirizzo della RAM (effettuata con i valori binari risultanti dal conteggio di ogni singolo 74F161) sia simultanea e non affetta da ritardi che potrebbero determinare la lettura o la scrittura nelle locazioni sbagliate. Ciò si comprende considerando che ogni contatore introduce un certo ritardo (detto tempo di propagazione) dal momento della ricezione dell'impulso di clock a quello in cui imposta di conseguenza le proprie uscite; se il ritardo si moltiplica per il numero di unità che può contare, già si ottiene un valore decisamente incrementato. Ora, se il contatore si connette in cascata con altri, i seguenti commuteranno la condizione delle proprie uscite con un ritardo sempre crescente, in funzione della distanza dal primo della catena; raggiunto un certo numero di conteggi, l'uscita più lontana non commuterà più in corrispondenza dell'impulso di clock, ma dopo, sul fronte opposto, oppure, peggio ancora, al successivo impulso. Ecco quindi che il contatore perderà un'unità. Siccome nel nostro circuito usiamo tre contatori a quattro bit ciascuno per costruire

sequenzialmente gli indirizzi per la memoria e sfruttiamo il fatto che le uscite di ciascun 74F161 si dispongono a rappresentare lo stato attuale del conteggio, se il terzo contatore perdesse un impulso di clock o si adeguasse con un ritardo significativo, nel momento in cui la PAL andrebbe a comandare la lettura o la scrittura l'operazione riguarderebbe una locazione sbagliata e non quella desiderata. Invece, con la tecnica adottata l'errore temporale non è possibile e tutti i 74F161 contano nello stesso momento.

A riguardo è opportuno spiegare che in realtà i contatori non vengono triggerati in parallelo, ma sincronizzati dallo stesso segnale di clock, dato in parallelo per evitare gli errori derivanti dalla Carry Out; ci spieghiamo meglio: ogni 74F161 riceve effettivamente il clock su comando della Carry Out del precedente, ma non da tale uscita. Quando arriva l'impulso dalla COUNT di un contatore, il seguente abilita la propria linea CLK a ricevere l'impulso dal filo comune COUNT. Quindi, se è vero che la Carry Out può essere in ritardo, è vero altresì che il clock è in comune e giunge con la stessa frequenza a tutti i contatori, quindi l'eventuale ritardo di commutazione viene diviso per lo stesso fattore corrispondente al conteggio e perciò, teoricamente, annullato.

Ancora, si osservi che i contatori sono presettabili, ovvero possono iniziare il conteggio da un numero il cui valore binario può essere definito mediante le linee P0, P1, P2, P3; nel nostro caso il conteggio deve avvenire dal minimo al massimo e quindi partire da zero, ed è perciò che i piedini 3, 4, 5, 6 sono posti fissi a zero logico.

Quanto alla composizione degli address, viene resa possibile dal fatto che ogni 74F161, raggiunto il valore limite (16) che può contare,

si azzerata e riprende da capo, intanto che il seguente conta anch'esso, sollecitato dal trigger ricevuto dalla Carry Out. Per fare un esempio, quando IC7A raggiunge per la prima volta il valore 16, le sue uscite assumono la condizione 1111; all'arrivo del successivo impulso di clock sulla linea COUNT, COUT dà un impulso a livello alto e fa avanzare di un passo il conteggio dell'IC8A, la cui uscita 0 si porta ad 1 logico. Naturalmente quando ciò avviene il primo contatore presenta già tutte le uscite a 0, perché si è automaticamente azzerato per riprendere il conteggio. Al successivo impulso sulla linea COUNT, IC7A presenta la situazione 1000 (bit 0÷3) mentre IC8A rimane fermo ad 1000 (sempre bit 0÷3); il corrispondente valore è 18.

Il bus dati della memoria, che poi è lo stesso dell'A/D converter, viene interfacciato con il computer mediante un multiplexer 74HC153, che trasferisce, ovvero rende seriali, gli otto bit su uno solo; in perfetto sincronismo con la linea READ e comandato da DSR e CLK1 in arrivo dalla porta parallela estratti dall'interfaccia optoisolata, il multiplexer colloca in fila, uno dopo l'altro, i singoli bit costituenti ciascun byte e li manda (tramite A e B) ai contatti 12 e 10 del connettore corrispondente. Notate che vengono utilizzate due linee perché il multiplexer è composto da due sezioni 4 su 1, ciascuna delle quali si occupa di quattro bit.

Detto ciò, non resta che analizzare l'interfaccia optoisolata che permette al computer di impartire le necessarie istruzioni all'hardware dell'oscilloscopio e a quest'ultimo di comunicargli i dati che esso richiede; consiste in cinque fotoaccoppiatori ad alta velocità di commutazione (di tipo 6N136) e una sorta di regolatore di tensione approntato per alimentare gli stadi dal lato del computer. Partiamo proprio da tale regolatore, che viene realizzato mediante due stadi: il primo usa un transistor PNP il cui collettore preleva corrente dall'insieme degli otto bit della parallela (mediante i diodi D8÷D15) e serve a far funzionare i led degli optoaccoppiatori IC19, IC20 e IC21. T1 funziona da interruttore statico e viene mandato in conduzione solamente quando la linea AUTO FEED della parallela si pone a zero logico, quindi soltanto quando il computer deve inviare il clock e i dati alla nostra interfaccia di misura; in ricezione, ossia quando l'interfaccia risponde al PC, AUTO FEED viene fatta tornare a livello alto in modo da lasciare a riposo T1 e non alimentare i predetti fotoaccoppiatori.

Si noti che per dare la necessaria corrente, il PC pone fissi a livello logico alto (5 V) i bit D3÷D7 della parallela e sfrutta anche gli impulsi inviati a D0÷D2; i diodi sono indispensabili ad evitare che le linee allo stato alto vengano cortocircuitate da quelle che si trovano a zero, cosa che

potrebbe danneggiare la porta della stampante. Lo stadio facente capo a T1 serve essenzialmente a interrompere l'assorbimento di corrente da parte di IC19, IC20 e IC21 quando l'interfaccia optoisolata deve trasmettere dati al PC; ciò perché i led dei fotoaccoppiatori consumano abbastanza e determinerebbero un eccessivo abbassamento del potenziale che alimenta gli stadi di uscita di IC18 e IC22. Questi ultimi possono, invece, restare sempre attivi, dato che consumano poco.

Quanto all'alimentazione degli stadi di uscita di IC18 e IC22, va detto che è distinta per fotodiodi e collettori dei fototransistor: la tensione per i fotodiodi viene ricavata dai contatti 6, 7, 8 (rispettivamente D4, D5, D6) del connettore a 25 poli, la cui corrente, quando sono a livello alto, confluisce da R50, R51, R52 sui piedini 8; quella per i collettori viene prelevata dal contatto 9 della parallela (bit dati D7) per IC22 e dall'8 (bit D6) per IC18.

Riepilogando, IC21 riceve dal bit D0 il segnale dati diretto alla nostra interfaccia di misura, IC20 accoppia D1 con la linea CLK1 (clock interno dell'interfaccia di misura) e IC19 riceve dal bit D2 l'RCK1. Riguardo ai fotoaccoppiatori preposti all'output, IC18 passa al computer (mediante il filo PE, cioè PAPER ERROR) i dati seriali della prima sezione del multiplexer (A) e IC22 pilota ACK (ACKNOWLEDGE) dandogli i restanti quattro bit di dati del ➤

per il

MATERIALE

L'oscilloscopio per PC presentato in queste pagine è disponibile in scatola di montaggio (cod. K8031) al prezzo di 144,00 Euro. Il kit comprende tutti i componenti, la basetta forata e segnaletta, il contenitore e le minuterie. Non sono compresi l'alimentatore e la sonda. L'oscilloscopio è anche disponibile già montato e collaudato (cod. PCS100A) al prezzo di 170,00 Euro. In questo caso la fornitura comprende anche l'alimentatore e una sonda.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-778112 ~ <http://www.futuranet.it>



presente il livello basso; se preferite, quando il led viene polarizzato direttamente il fototransistor di uscita va in saturazione. Adesso che è stato spiegato come funzionano i 6N136, va notato un particolare dell'IC18: per far illuminare il suo led occorre che la linea END_COUNT sia costantemente ad uno logico; essa viene gestita dalla PAL per effettuare la trasmissione

alternata dei due pacchetti di 4 bit, ora su PE (contatto 12 della parallela) ora su ACK (10).

Realizzazione pratica

L'hardware dell'oscilloscopio si costruisce partendo dall'apposito circuito stampato a doppia faccia, quindi disponendo su di esso i componenti, a cominciare da quelli a più basso profilo; per tutti gli integrati conviene prevedere il montaggio su zoccolo, che consente una rapida sostituzione in caso di guasto e mette al riparo dal rischio di danneggiamento per surriscaldamento in fase di saldatura. Il regolatore IC23 va dotato di un dissipatore di calore avente non più di 15 °C/W di resistenza termica; volendo, può essere appoggiato dalla parte dell'aletta metallica al retro del contenitore, sempre che quest'ultimo non sia in plastica.

Per il collegamento della sonda di misura bisogna disporre sullo stampato una presa BNC; una seconda presa, ma di tipo plug coassiale, serve per l'alimentazione. Il circuito si completa inserendo a fondo e stagnando (anche nelle alette di fissaggio) il connettore DB-25 maschio. Sistemati tutti i componenti, conviene dare uno sguardo all'insieme verificando, con i piani di montaggio, che ogni cosa sia al proprio posto; se tutto è ok, conviene procurarsi

un contenitore adatto, possibilmente metallico perché fa da schermo elettromagnetico e protegge il circuito da interferenze che falserebbero le letture e sporcherebbero le forme d'onda visualizzate nello schermo, impedendo all'operatore di capire la natura del disturbo. Si può anche usare un contenitore plastico, ma in tal caso conviene appoggiare sul fondo una lamina di rame o di alluminio da collegare elettricamente alla massa di alimentazione (pista vicina alla presa plug); la lamina dovrà coprire almeno la zona contenente i relé, il BNC, gli operazionali e l'A/D converter. Un secondo schermo andrà incollato all'interno del coperchio, in modo che risulti in linea con quello sottostante e copra i predetti elementi circuitali; anch'esso andrà collegato con un filo alla massa di alimentazione. Resta inteso che il lato saldature e i componenti posti dall'altra faccia non dovranno entrare in contatto con le lamine, altrimenti ne risulterebbero dei cortocircuiti. Dal pannello frontale del contenitore dovranno affacciarsi il BNC e il led che indica la presenza dell'alimentazione; dal retro bisognerà fare uscire il connettore DB-25 e rendere accessibile (praticando un foro di diametro adeguato) il plug dell'alimentatore.

Chiuso il contenitore, la nostra interfaccia è pronta per l'uso; basta collegarla al PC utilizzando un cavo parallelo maschio/femmina, del quale siano collegati tutti i 25 contatti. Nella prossima puntata spiegheremo come utilizzarla in ambiente Windows mediante l'apposito programma di gestione e pubblicheremo, per lo stesso programma, la procedura di preparazione di una macro che permette di importare in un foglio di lavoro di Microsoft Excel i dati relativi alle misure effettuate. Descriveremo, inoltre, come modificare la DLL che sta alla base del funzionamento del software per personalizzare il proprio programma.

multiplexer, cioè quelli uscenti da B. I fotoaccoppiatori usati sono costituiti da un led d'ingresso e l'insieme di un fotodiodo e di un transistor all'uscita; tale soluzione permette tempi di commutazione molto ridotti rispetto a quelli garantiti dal tipico accoppiamento led-fototransistor (usato, per esempio, nei 4N25 o 4N35): infatti il fototransistor ha il difetto di impiegare un intervallo relativamente lungo per passare dalla conduzione all'interdizione, e ciò perché le cariche minoritarie accumulate nella base durante la conduzione sono molte e lente a ricombinarsi. Per aumentare la velocità di un fototransistor occorrerebbe collegare alla base del fototransistor una resistenza che permettesse di far fluire la corrente all'interdizione e accelerare la ricombinazione, ma ciò, in conduzione, significherebbe abbassare il CTR (Current Transfer Ratio) e quindi aumentare la richiesta di corrente da parte del led.

Il 6N136 risulta veloce e il suo led di ingresso assorbe poca corrente, sia perché il fotodiodo è più veloce del fototransistor nel passaggio dalla conduzione all'interdizione, sia per il fatto che collegandolo in serie alla base di un transistor viene aumentata decisamente la corrente erogabile dal collettore e quindi dall'uscita.

Con l'attuale configurazione, gli optoisolatori presentano in uscita lo zero logico quando all'ingresso è

Protocollo seriale Velbus: dalla teoria alla pratica

di Corrado Rossi

Esaminiamo il protocollo seriale Velbus in dettaglio.

Aiutandoci con un programma di test realizzato appositamente, analizziamo la struttura delle stringhe di comando focalizzando l'attenzione sulla sintassi e sul significato di ogni bit.

Vediamo quindi alcuni esempi pratici applicati alle funzioni domotiche del nostro monolocale d'esempio.

Nei precedenti articoli dedicati ai sistemi domotici Velbus abbiamo visto come può essere realizzato un impianto di automazione domestica dotato di funzioni molto complesse, ma accessibili a chiunque con grande facilità. Quanti hanno letto tali articoli, ricorderanno che, partendo dal disegno topografico di un ipotetico appartamento monolocale, avevamo posizionato (e collegato) nei vari punti di controllo i dispositivi Velbus con cui dovevamo comandare lo stato delle luci e l'apertura e chiusura delle tapparelle. Avevamo poi analizzato, passo dopo passo, la procedura per la programmazione

manuale con la quale si erano definite le mansioni svolte da ogni dispositivo della rete. Concludevamo quindi con l'analisi del programma di configurazione VelbusLink (Ver. 4.4), un applicativo da installare su di un personal computer, a sua volta collegato alla rete domotica tramite un circuito di adattamento da USB (VMB1USB) a CAN Bus su RS485 (l'interfaccia elettrica utilizzata dai dispositivi Velbus). Tale programma ci ha permesso sia di configurare la rete con maggiore semplicità rispetto alla procedura manuale, sia di rilevare e visualizzare il traffico dati presente sul CAN Bus. A questo punto, ➤

Note relative a VMB1USB e VMB1RS

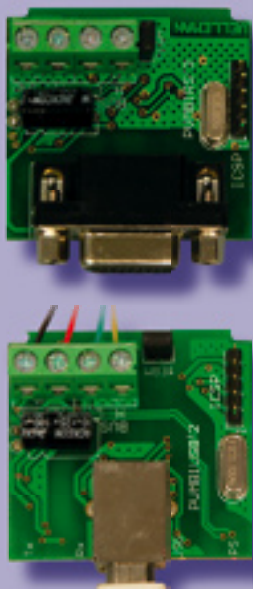
Una caratteristica molto importante dei sistemi domotici è rappresentata dalla possibilità di interagire con altri sistemi, siano essi esistenti o futuri. In questo modo la naturale evoluzione tecnologica dei sistemi non obbliga ad alcun rifacimento delle installazioni esistenti. In sintesi, è conveniente che un sistema domotico disponga sempre di una via di comunicazione con l'esterno. Nel caso del sistema Velbus, tale accesso è costituito dalle due schede VMB1RS e VMB1USB, dispositivi che dispongono sia di una interfaccia CAN-Bus che di una porta di comunicazione seriale (VMB1RS) o USB (VMB1USB); entrambe le schede sono dotate di un microcontrollore che converte le stringhe di dati in transito sul bus bifilare da "formato CAN-Bus" a "formato Velbus seriale" e viceversa.

Dal punto di vista funzionale, le due schede differiscono solo per il driver software da installare sul computer: il PC, infatti, comunica con esse aprendo la comunicazione seriale su una COM Port, la quale può essere hardware (nel caso della VMB1RS) o creata virtualmente dal driver (quando si usa la scheda VMB1USB).

Da quanto detto, quindi, è possibile trarre una conclusione molto importante: la versione seriale può essere utilizzata con qualsiasi sistema elettronico dotato di un programma (sia esso un PC o una piattaforma embedded) purché questi disponga di una porta seriale. La versione USB, invece, è più indicata in ambito PC, dove la porta seriale sta scomparendo.

In ogni caso occorre prestare attenzione a tre particolari importanti, indipendentemente dal modello di scheda scelto per far dialogare il Velbus con l'applicativo hardware/software:

- 1) Le schede non hanno indirizzo: infatti non sono dotate dei preselettori binari rotativi di cui dispongono tutti gli altri dispositivi Velbus. Questo significa che esse sono invisibili al protocollo CAN-Bus sul doppino RS485.
- 2) Poiché sono elettricamente collegate al bus, dispongono entrambe di un ponticello con cui inserire la resistenza di terminazione ed adattamento di impedenza. Questa caratteristica è comune a tutti i dispositivi Velbus.
- 3) Necessitano entrambe della doppia alimentazione in quanto la linea CAN-Bus e la porta di comunicazione sono isolate galvanicamente. Riferendosi al lato PC/micro, nella versione VMB1RS tale tensione è prelevata dalle linee DTR (forzato basso) e RTS (forzato alto). La VMB1USB, invece, trae alimentazione direttamente dalla porta USB del computer.



per completare la descrizione del sistema Velbus, non rimane che comprendere il funzionamento del protocollo seriale utilizzato dal programma VelbusLink, protocollo che analizzeremo dettagliatamente, avvalendoci anche del supporto di un nuovo programma di test sviluppato espressamente per questo scopo. Grazie a tale programma (EasyVelbus), potrete verificare sperimentalmente tutti gli esempi pratici forniti a complemento delle spiegazioni teoriche. Vi anticipiamo subito una cosa molto importante: nonostante ogni dispositivo Velbus disponga di comandi specifici, la sintassi di tali comandi è unica. Quindi, una volta imparato a comporre ed inviare una stringa di comando, sarete in grado di interagire facilmente con qualsiasi

dispositivo Velbus presente nella vostra applicazione reale.

Il protocollo seriale

Da quanto detto negli articoli precedenti, sappiamo che i dispositivi Velbus comunicano tra loro per mezzo di un bus bifilare di tipo RS485, sul quale transitano i dati secondo una codifica di tipo CAN Bus, standard di cui abbiamo già pubblicato un corso molto dettagliato nei fascicoli precedenti. Sappiamo inoltre che Velleman ha realizzato due schede di interfaccia, una seriale ed una USB, per far interagire la rete domotica con sistemi elettronici differenti: in particolare, nell'articolo pubblicato sul fascicolo 116 abbiamo usato l'interfaccia VMB1USB per controllare la rete via computer

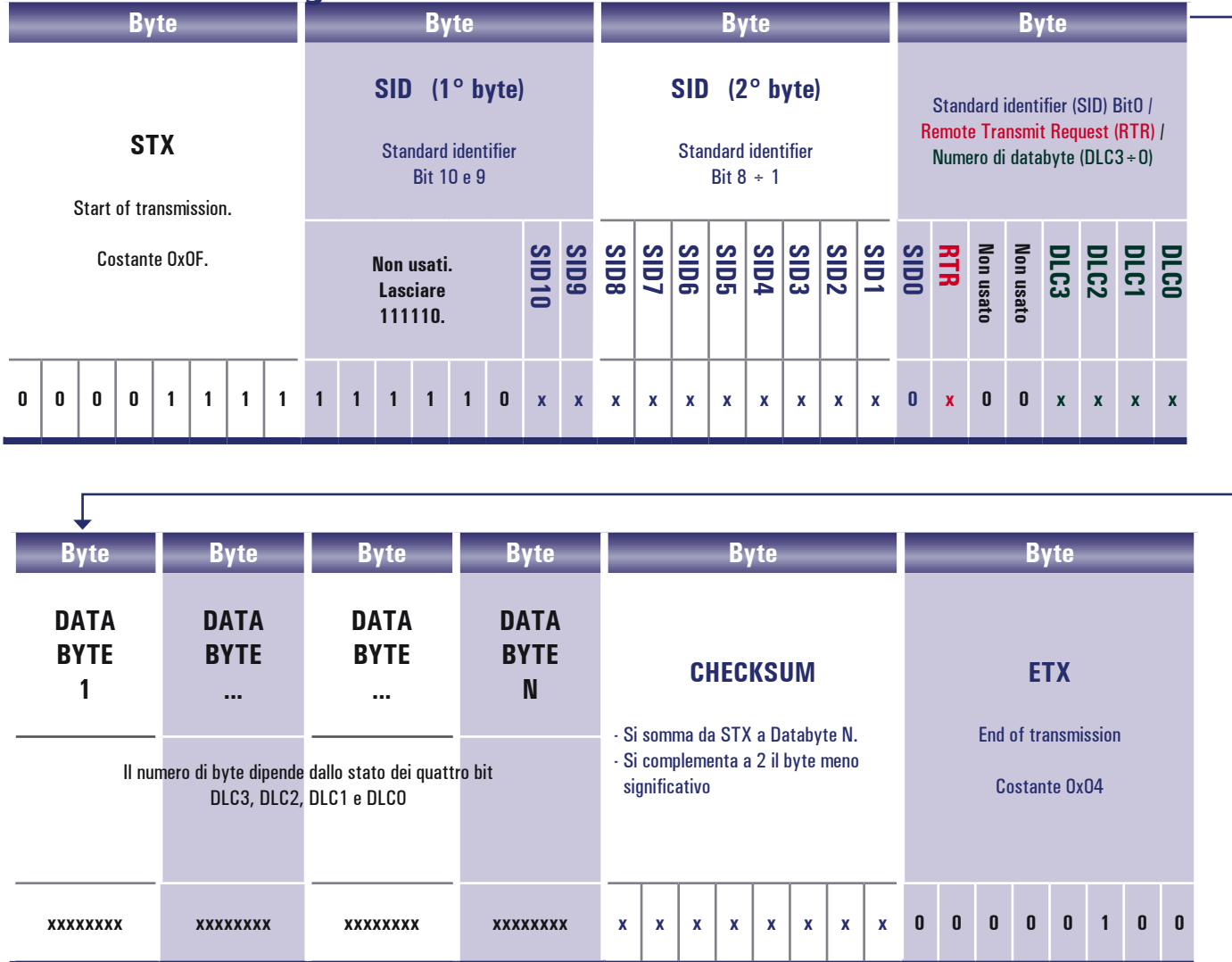
con il programma VelbusLink. Focalizziamo a questo punto l'attenzione su tale scheda, cercando di intuire quale funzione svolge. Ci avete pensato? Esatto, è proprio così: la scheda svolge la funzione di "traduttore", nel senso che essa "ascolta" il traffico dati presente sulla rete Velbus, lo intercetta senza alterarne il contenuto, quindi lo ripropone sull'interfaccia USB in un formato più semplice da gestire. Analogamente, essa ascolta tutto ciò che il computer le invia tramite la porta USB: se il pacchetto di dati ricevuto è corretto, sia come struttura che come sintassi, la scheda lo traduce e lo trasferisce sul bus dati RS485.

Quanto detto per la scheda VMB1USB vale anche per la VMB1RS: dal punto di vista funzionale, infatti, questi dispositivi differiscono solamente per il "driver" software, usato dal PC per creare una porta seriale "virtuale" tramite la quale comunica con la scheda USB. Comunque, indipendentemente dalla scheda di interfaccia in uso, ogni dato in transito sul doppino viene riproposto sulla porta di comunicazione con il PC e viceversa secondo il formato riportato nella pagina successiva.

Prima di iniziare a descrivere la struttura del protocollo, occorre avere chiaro un concetto molto importante: ogni volta che parliamo di "linea di comando" o di "stringa di dati" ci riferiamo ad una sequenza di byte contenente, oltre alle informazioni di servizio (inizio stringa, fine stringa, richiesta di conferma di ricezione, indirizzo, numero di byte ecc., dettagliate in seguito), i "dati" (leggasi "i valori numerici") che descrivono il "comando" vero e proprio ed i relativi "parametri".

A questo punto scopriamo insieme come è stato concepito il protocollo Velbus.

Come visibile nella tabella della pagina accanto, ogni linea di comando è suddivisa in più byte (alcuni di va-



lore fisso ed altri invece variabili in funzione delle operazioni da eseguire) e può assumere una lunghezza compresa tra 6 e 14 byte.

Il primo byte della sequenza è l'STX (Start Of Transmission) ed ha valore fisso a 0x0F (decimale 15). Il suo unico compito è di identificare l'inizio della stringa di dati, altrimenti non riconoscibile.

All'STX fa seguito il primo byte SID (Standard Identifier), in cui i bit da 7 a 3 sono fissi a "1" logico, il bit 2 è a "0" logico, ed infine i bit 1 e 0 (SID10 e SID9) definiscono la priorità della stringa. SID 10 e 9 possono assumere attualmente solo i valori "11" (priorità bassa) e "00" (priorità alta). Quindi l'escursione del primo byte SID è compresa tra 0xF8 e 0xFB (da 248 a 251 se

espresso in decimale), benché 0xF9 e 0xFA non siano ancora utilizzati.

Segue il secondo byte SID (terzo della serie), i cui bit sono nominati da SID8 (bit più significativo) a SID1 (bit meno significativo). Questo byte identifica sempre l'indirizzo di un dispositivo: nel caso in cui il messaggio sia ricevuto dal PC, l'indirizzo si riferisce al dispositivo che ha generato il messaggio. Se invece il messaggio è generato dal PC (o da un altro dispositivo che deve inviare un comando), il byte contiene l'indirizzo del dispositivo destinatario.

Il quarto byte non ha un nome, ma contiene due informazioni molto importanti: la richiesta di conferma di ricezione di un comando ed il numero di byte necessari per definire il comando e i suoi parametri, nel-

l'azione in corso.

La richiesta di trasmissione è utilizzata prevalentemente quando un dispositivo (ad esempio il gruppo pulsanti VMB1P6) genera un comando per un altro dispositivo (ad esempio il relé VMB1RY). Tale richiesta è effettuata forzando l'RTR (bit 6, Remote Transmit Request) a "1" logico.

Il numero di byte necessari per descrivere il comando è definito dai bit da 3 a 0, nominati DLC3 - DLC2 - DLC1 -DLC0. A titolo di esempio, se tali bit hanno valore 0001 (decimale 1), ciò significa che dopo il quarto byte segue un byte di dati. Analogamente, la condizione logica 1000 (decimale 8), indica che i byte dati sono 8. Tenete presente che quest'ultimo è il massimo numero di

CHECKSUM: cos'è, come si calcola e come si usa

Il CHECKSUM è un campo numerico utilizzato spessissimo dai protocolli di trasmissione dati per consentire al destinatario di un messaggio o comando di verificare la correttezza della stringa di dati ricevuta. Esso viene calcolato dal mittente e ricalcolato dal destinatario, secondo una formula specifica nota ad entrambi.

Analizziamo il semplice esempio di struttura a sette byte riportato qui di seguito, in cui DATA1, 2 e 3 rappresentano le informazioni da recapitare:

INIZIO STRINGA	PARAMETRI	DATA1	DATA2	DATA3	CHECKSUM	FINE STRINGA
-------------------	-----------	-------	-------	-------	----------	-----------------

Il calcolo del CHECKSUM deve sicuramente tenere in considerazione il valore dei campi DATA e PARAMETRI però, in alcuni casi quali il nostro, tiene in considerazione anche il campo che identifica l'inizio della stringa.

Nel caso specifico del protocollo seriale Velbus, il CHECKSUM ha dimensione di 1 byte, è posizionato immediatamente dopo l'ultimo DATA BYTE e si ottiene applicando la procedura seguente:

- Si esegue la somma dei byte compresi tra l'STX (inizio stringa) e l'ultimo DATA BYTE prima del campo CHECKSUM. Sia l'STX che l'ultimo DATA BYTE sono inclusi nel calcolo;
- Se il risultato ottenuto ha dimensione superiore ad 1 byte, si estrae solo il byte meno significativo (LSB);
- Si esegue il complemento a 2 del byte (si inverte ogni bit e si incrementa il risultato di "1").

Il principio di utilizzo è semplice: il mittente del messaggio sa esattamente cosa deve inviare, quindi è in grado di calcolare il CHECKSUM e di aggiungerlo in un punto noto nella sequenza di dati. Il destinatario, invece, non sa cosa può ricevere, ma sa esattamente che quanto riceve è delimitato dai campi di inizio e fine. Quindi può estrarre sia il CHECKSUM che tutti i dati, ricalcolare il CHECKSUM in locale, ed infine confrontarlo con quello ricevuto. Se i due valori numerici sono identici, la sequenza di dati è stata ricevuta correttamente, quindi viene convalidata e processata.

Vediamo ora l'esempio seguente, il quale rappresenta una sentenza rilevata durante il normale funzionamento di EasyVelbus:

15	251	6	2	246	4	244	4
----	-----	---	---	-----	---	-----	---

Da quanto visto nella pagina precedente, <15> delimita l'inizio della stringa (STX) mentre <4> ne determina la fine. Secondo il protocollo, il CHECKSUM si trova prima del carattere di fine stringa, quindi è <244>. Applichiamo la procedura:

- Somma numerica dall'STX all'ultimo DATA BYTE:
 $15 + 251 + 6 + 2 + 246 + 4 = 524$, che in esadecimale equivale a $0x20C$
- Estrazione LSB: il risultato è $0x0C$
- Complemento a 2 di $0x0C$ (si nega ogni bit e si somma "1"):
 - $0x0C$ equivale a 00001100 in binario. La negazione diventa 11110011 .
 - Sommando 1 si ottiene 11110100 , equivalente a 244 in decimale.

Da quanto esposto si può comprendere che il CHECKSUM, normalmente utile, è addirittura indispensabile quando i dati sono strutturati in sequenze di byte molto lunghe.

byte dati supportato dal protocollo. I bit 7 (SID0), 5 e 4 sono fissi a 0 in quanto non utilizzati.

Indipendentemente dal numero di data byte contenuti nella stringa, il primo byte successivo all'ultimo data byte è il Checksum: esso rappresenta un valore numerico ricavato dai valori di tutti i byte precedenti secondo una formula precisa. Il Checksum è molto importante perché permette a un dispositivo di effettuare una verifica della stringa

appena ricevuta per avere la certezza che i dati sono corretti. La verifica della stringa avviene in due fasi: nella prima fase, il dispositivo che invia la stringa genera il checksum partendo dai dati che deve inviare. Nella seconda fase, invece, il dispositivo che riceve i dati (compreso il checksum) ricalcola il checksum localmente partendo dai dati ricevuti (dallo start byte fino all'ultimo data byte), quindi lo confronta con quello contenuto nello stesso pacchetto

dati: se i due valori sono uguali, il pacchetto dati è stato ricevuto correttamente; affinché la procedura funzioni, mittente e destinatario devono usare la stessa formula di calcolo. In questo caso la formula è piuttosto semplice: si esegue la somma logica di tutti i byte compresi tra l'STX e l'ultimo databyte (inclusi), quindi si estrae dal risultato solo il byte meno significativo e si esegue su di esso il complemento a 2 (funzione in cui si esegue l'inversione logica di ogni

singolo bit e si somma una unità "1" al risultato ottenuto). Data l'importanza del checksum, abbiamo voluto dedicare ad esso il riquadro della pagina precedente.

Il byte l'ETX (End Of Transmission), che ha valore fisso 0x04 (decimale 4), determina la fine della stringa.

Avete sicuramente constatato che la struttura del protocollo, in linea di massima, è molto semplice: infatti, una volta noto che la stringa inizia con 0x0F e termina con 0x04, che il quarto byte contiene la richiesta di conferma di ricezione ed il numero di databyte e che il penultimo byte è il checksum, estrarre da essa i campi relativi al comando vero è proprio è abbastanza facile. La parte complicata, invece, è costituita dal comando "vero e proprio" contenuto nei databyte che, come visto, può avere una lunghezza compresa tra "0" e "8" byte. Va da sé che maggiore è la lunghezza, più complicato è il comando, indipendentemente dal fatto che esso sia ricevuto o inviato.

Ultimo ma non ultimo, fino ad ora abbiamo parlato di "comando" (al singolare), ma da questo momento è opportuno parlare di "comandi", visto che ogni singolo dispositivo Velbus ne può riconoscere e generare molti distinti a seconda delle istruzioni.

Facciamo un semplice esempio: il primo ingresso di un modulo VMB6IN attiva per 30 secondi il

relé del modulo VMB1RY.

Premuto il pulsante, il VMB6IN invia al relé la richiesta di attivazione: a questo punto VMB1RY genera una stringa di risposta se il bit RTR del comando appena ricevuto si trova a 1.

Quindi invia a sua volta un comando al VMB6IN chiedendogli di far lampeggiare il led corrispondente al primo pulsante per tutto il periodo in cui il relé è eccitato.

Come avvenuto per il relé, l'RTR a livello logico 1 forza il modulo pulsanti a confermare la ricezione del dato. Passati i 30 secondi, il modulo VMB1RY genera due sentenze: con la prima informa la rete di avere disattivato il relé e con la seconda chiede al pulsante di spegnere il led.

Qualora i due comandi avessero il bit RTR a 1, la scheda pulsanti risponderebbe nuovamente con due messaggi di conferma di ricezione. L'esempio appena fatto rende l'idea di quanto l'insieme delle istruzioni contenute nei databyte nelle stringhe di comando sia complesso, nonostante il protocollo in sé sia molto semplice. Per questo motivo, non potendo analizzare uno per uno tutti i comandi e relativi parametri, abbia-



mo realizzato un programma specifico in grado di:

- tenere traccia di ogni informazione in transito sul bus,
- analizzare ogni bit di ogni stringa;
- generare delle stringhe di comando ed inviarle.

Tale programma, chiamato EasyVelbus.exe, è scaricabile dal sito della rivista (www.elettronica.in.it).

L'applicativo EasyVelbus

Il sistema Velbus ha riscosso tanto successo che molti utenti (sia installatori che privati), si sono voluti cimentare nella realizzazione di applicativi di controllo su PC. Per facilitare lo sviluppo di tali programmi, Velleman ha reso disponibile sul proprio sito (www.velleman.be) una serie di applicativi di test e relativo codice sorgente, tutti realizzati in ambiente "Visual C .NET" e incen-



RM ELETTRONICA SNC

vendita componenti elettronici
rivenditore autorizzato:

 **FUTURA ELETTRONICA**

 **ELETTRONICA**

G.P.E.



Via Val Sillaro, 38 - 00141 ROMA - tel. 06/8104753

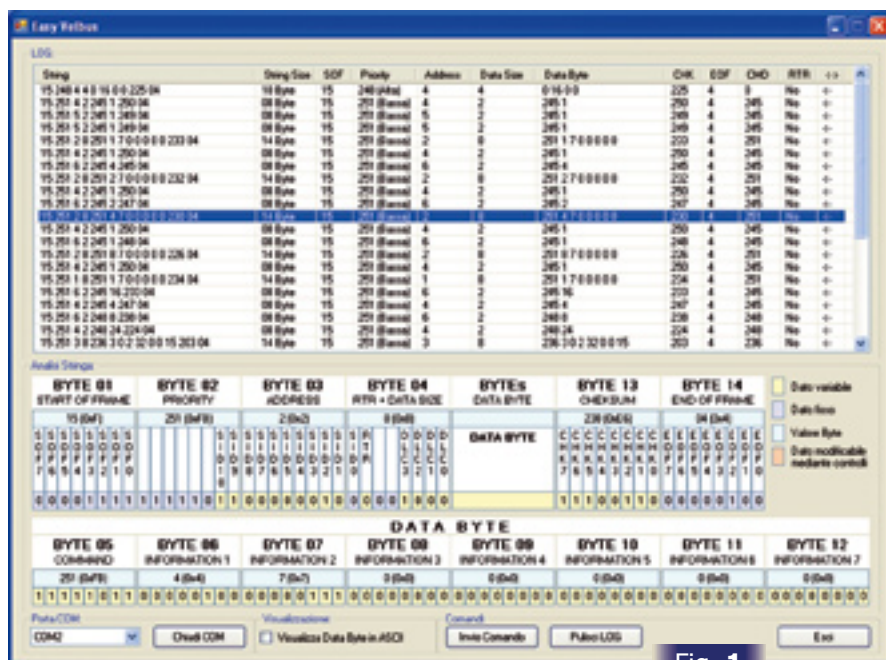


Fig. 1

- “Visualizza DataByte in ASCII” (funzione utile quando la stringa contiene il nome dei dispositivi, identificato da un testo);
- il checksum;
- il terminatore di stringa;
- il comando estratto (contenuto nel primo databyte);
- lo stato dell’RTR;
- la provenienza del messaggio (dal bus o dal PC).

Il registro degli eventi può essere ripulito cliccando sul pulsante “Pulisci LOG”, posto nel riquadro “Comandi”, insieme ai pulsanti “Invia Comando” ed “Esci”.

Per analizzare un messaggio occorre selezionarlo con un clic del mouse: in questo modo il programma lo scompone e lo ripropone nel frame “Analisi Stringa” in cui potete vedere lo stato di tutti i byte e di ogni singolo bit nei tre formati “dato” (esadecimale, decimale e binario). I bit in campo azzurro sono fissi mentre quelli in campo giallo sono variabili. Quelli in campo arancio, invece, vengono modificati tramite i controlli del programma.

Il campo “DATA BYTE”, compreso nella prima linea, è stato espanso e scomposto in ogni singolo bit nella seconda linea. In questo modo è possibile tenere sotto controllo l’intera stringa di dati.

Da quanto emerso fino ad ora, il programma, essenzialmente, visualizza il traffico sul CAN Bus (funzione chiamata “Sniffer”); come avevamo già anticipato, però, esso è anche in grado di comporre ed inviare delle stringhe di comando, controllandone la lunghezza ed aggiungendo il checksum. Per fare ciò si deve cliccare sul pulsante “Invia Comando” presente nella finestra principale (Fig. 1). Appare la finestra di composizione delle stringhe (Fig. 2), utilizzata soprattutto per verificare gli esempi esposti nel paragrafo successivo. Prestate attenzione a un particolare molto importante:

trati su una libreria chiamata “vel-*leman.velbus.dll*”, compatibile con l’ambiente di sviluppo “VisualBasic 2005.NET” (“o superiore): noi stessi abbiamo utilizzato tale libreria per realizzare il programma EasyVelbus in ambiente VisualBasic 2005.NET. Qualora si volesse utilizzare la sola libreria, tralasciando gli applicativi di esempio, è sufficiente creare un nuovo progetto in ambiente VisualBasic ed includere la DLL tramite il *menu strumenti*, utilizzato per definire le specifiche del progetto.

In Fig. 1 è visibile l’interfaccia grafica principale di EasyVelbus: essa fornisce molte informazioni, per cui potrebbe sembrare complessa, ma non preoccupatevi più di tanto; ora la vediamo insieme, così vi rendete conto che è molto intuitiva e facile da utilizzare. Dopo aver installato ed avviato il programma, occorre scegliere la porta seriale per dialogare con la scheda USB (ricordatevi che essa viene vista come una seriale virtuale): per fare ciò è sufficiente selezionare la COM tramite la combo box in basso a sinistra, quindi cliccare sul tasto “Apri COM” (il nome cambia in “Chiudi COM”); a questo punto il programma instaura la comunicazione sulla seriale pre-

scelta con il formato dati “38.400 baud - parità N - 8 bit di dati - 1 bit di stop”; esso, inoltre, disattiva il DTR e attiva l’RTS, condizione questa necessaria per poter utilizzare EasyVelbus anche con la scheda VMB1RS.

Quando il programma è in comunicazione con la rete Velbus, qualsiasi stringa di dati in transito (più in generale, qualsiasi “evento”) viene rilevata e presentata sulla tabella (chiamata “Log”) che occupa la parte superiore dell’interfaccia grafica e non è modificabile dall’utente. Tale tabella è suddivisa in colonne e visualizza, da sinistra a destra:

- la stringa completa, inclusi i byte STX e ETX (inizio e fine);
- il numero di byte da cui è composta;
- l’identificativo di inizio stringa;
- la priorità della stringa;
- l’indirizzo mittente o destinatario;
- il numero di databyte;
- i databyte trasportati, visualizzabili in ASCII mettendo il segno di spunta sulla checkbox in basso



Fig. 2

Nell'articolo pubblicato sul fascicolo 116 abbiamo spiegato come configurare la rete Velbus tramite computer utilizzando l'applicativo VelbusLink versione 4.4. Tale versione era la più recente disponibile quando abbiamo preparato l'articolo. Avevamo però anticipato che il VelbusLink è un programma di configurazione e test in continuo sviluppo e miglioramento, per cui era plausibile che versioni di programma più aggiornate avrebbero potuto differire, per aspetto e funzionalità, da quello usato da noi, pur rimanendo compatibile con le funzionalità base.

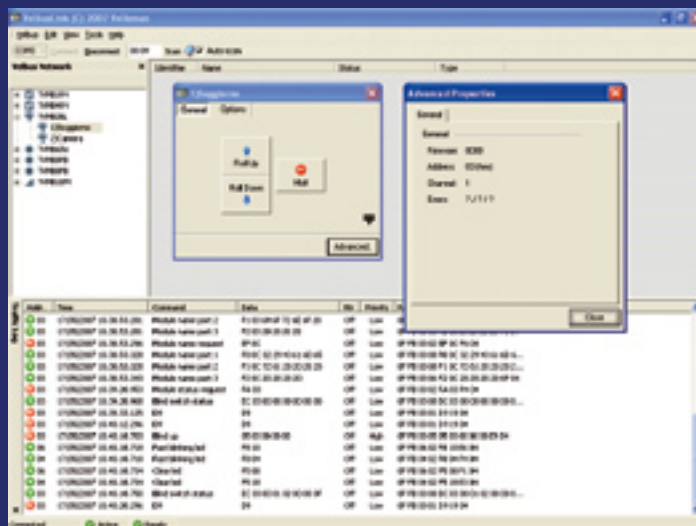
La versione 5.1, scaricabile attualmente dal sito www.velleman.be, mantiene le stesse funzionalità supportate dalla versione 4.4: consente quindi di effettuare la programmazione, il backup e la riprogrammazione delle configurazioni di tutti i dispositivi, permette di analizzare sia la struttura della rete che il suo traffico dati. Però presenta alcune migliorie che hanno aumentato notevolmente sia l'efficienza del programma che la facilità di utilizzo.

Innanzitutto l'interfaccia grafica è stata notevolmente migliorata: le frecce di colore differente nella finestra di analisi dei dati in transito permettono di capire più facilmente se i dati partono dal PC (arancione) o da un dispositivo Velbus (verde). Anche la presentazione dei dati in formato testo e esadecimale è stata migliorata, poiché ora è molto più ricca di dettagli. È stata inserita la possibilità di scegliere la gamma di indirizzi su cui effettuare la scansione che per esempio, nella foto a fianco, è stata limitata tra 00 e 09.

Questa funzione è particolarmente utile per due ragioni: alleggerisce la finestra del traffico dati (che diventa più comprensibile) e aumenta la velocità nell'esecuzione delle operazioni.

Ora è anche possibile disattivare la scansione automatica, funzione utile soprattutto se si vuole interagire con un numero limitato di dispositivi senza scollegare gli altri dalla rete.

La miglioria più importante, però, non è così evidente: ora, infatti, l'applicativo è molto più stabile mentre in precedenza generava occasionalmente degli errori di "runtime", specialmente operando su COM Port virtuale.



per rendere semplice il programma, abbiamo scelto di gestire i comandi in formato decimale, mentre nei manuali dei dispositivi viene usato il formato esadecimale (talvolta il binario). Ciò significa che i codici dei comandi, ricavati dai manuali, devono essere convertiti da esadecimale a decimale. Anticipiamo un esempio solo per chiarire questo punto, ipotizzando di voler verificare se l'indirizzo 0xFF, equivalente a 255 in decimale, è assegnato a qualche dispositivo. Da manuale si ricava che il comando relativo (Module Type Request) ha priorità bas-

sa (SID10 e 9 = 11), RTR a "1" e nessun "databyte". Quindi la finestra di composizione si presenta come la Fig. 3: notate l'indirizzo 0xFF in campo giallo, inserito manualmente, e, come dicevamo, espresso in decimale. È necessario agire nello stesso modo con qualsiasi campo variabile, sia esso inserito manualmente (indirizzo e dati) o generato dai controlli del programma, come nel caso di priorità, numero di databyte e checksum (in arancione). Siamo convinti, comunque, che la conversione non comporterà alcuna difficoltà. Relativamente ai comandi disponibili, non abbiamo voluto inserirli in un database per due ragioni: innanzitutto essi sono ben descritti nei manuali di ogni dispositivo Velbus, ed in secondo luogo il programma sarebbe diventato una copia del VelbusLink.



Fig. 3

Esempi Pratici

Dopo aver analizzato la struttura e la sintassi del protocollo e preso confidenza con il programma di test, è il momento di iniziare con gli esempi pratici, tenendo presente che un comando è sempre associato ad una serie di parametri. Ricordatevi, inoltre, che dovete usare la finestra di composizione dei comandi.

Nel corso dell'articolo abbiamo già detto che ogni modulo Velbus ha delle funzioni specifiche e di conseguenza un proprio set di comandi (sia in ricezione che in trasmissione). Però non avevamo ancora detto che molti comandi sono comuni a tutti i moduli (ad esempio le richieste di nome e tipo di dispositivo). Inoltre i dispositivi simili accettano gli stessi comandi, differenti solo per i parametri. In particolare, ➤

Tabella 1

COMANDO	AZIONE
PC ---> 0F FB 01 40 B5 04	Il PC interroga l'indirizzo 1.
PC < --- 0F FB 01 03 FF 02 FF F2 04	Messaggio di risposta composto da 3 byte dati: FF 02 FF FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 02 = ONE_CHANNEL_RELAY_MODULE_TYPE (0x02), il modulo è un VMB1RY FF = il relé è impostato come: - F (primo nibble): Dual Timer forzato in bistabile - F (secondo nibble): Relé forzato in bistabile.
PC ---> 0F FB 02 40 B4 04	Il PC interroga l'indirizzo 2
PC < --- 0F FB 02 06 FF 08 FF FF FF FF EB 04	Messaggio di risposta composto da 6 byte dati: FF 08 FF FF FF FF FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 08 = 4_CHANNEL_RELAY_MODULE_TYPE (0x08), il modulo è un VMB4RY FF FF FF FF = 1 relé sono tutti impostati come - F (primo nibble): Dual Timer forzato in bistabile - F (secondo nibble): Relé forzato in bistabile.
PC ---> 0F FB 03 40 B3 04	Il PC interroga l'indirizzo 3.
PC < --- 0F FB 03 03 FF 09 00 E8 04	Messaggio di risposta composto da 3 byte dati: FF 09 00 FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 09 = TWO_CHANNEL_BLIND_MODULE_TYPE (0x09), il modulo è un VMB2BL 00 = entrambi i canali hanno un time-out di 15 secondi
PC ---> 0F FB 04 40 B2 04	Il PC interroga l'indirizzo 4.
PC < --- 0F FB 04 05 FF 05 00 00 00 E9 04	Messaggio di risposta composto da 5 byte dati: FF 05 00 00 00 FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 05 = 6_CHANNEL_input_MODULE_TYPE (0x05), il modulo è un VMB6IN 00 = nessun led acceso a luce fissa 00 = nessun led lampeggia lentamente 00 = nessun led lampeggia velocemente
PC ---> 0F FB 05 40 B1 04	Il PC interroga l'indirizzo 5.
PC < --- 0F FB 05 05 FF 01 00 00 00 EC 04	Messaggio di risposta composto da 5 byte dati: FF 01 00 00 00 FF = COMMAND_MODULE_TYPE (0xFF), modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 05 = 6_CHANNEL_INPUT_MODULE_TYPE (0x05), il modulo è un VMB6IN 00 = nessun led acceso a luce fissa 00 = nessun led lampeggia lentamente 00 = nessun led lampeggia velocemente
PC ---> 0F FB 05 40 B1 04	Il PC interroga l'indirizzo 5.
PC < --- 0F FB 05 05 FF 01 00 00 00 EC 04	Messaggio di risposta composto da 5 byte dati: FF 01 00 00 00 FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 01 = 8_PUSH_BUTTON_MODULE_TYPE (0x01), il modulo è un VMB8PB 00 = nessun led acceso a luce fissa 00 = nessun led lampeggia lentamente 00 = nessun led lampeggia velocemente
PC ---> 0F FB 06 40 B0 04	Il PC interroga l'indirizzo 6.
PC < --- 0F FB 06 05 FF 01 00 00 00 EB 04	Messaggio di risposta composto da 5 byte dati: FF 01 00 00 00 FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 01 = 8_PUSH_BUTTON_MODULE_TYPE (0x01), il modulo è un VMB8PB 00 = nessun led acceso a luce fissa 00 = nessun led lampeggia lentamente 00 = nessun led lampeggia velocemente
PC ---> 0F FB 07 40 AF 04	Il PC interroga l'indirizzo 7.
PC < --- 0F FB 07 07 FF 07 03 0F 81 07 02 46 04	Messaggio di risposta composto da 7 byte dati: FF 07 03 0F 81 07 02 FF = COMMAND_MODULE_TYPE (0xFF), un modulo sta rispondendo ad una scansione, o richiesta di tipo di modulo. 07 = DIMMER_MODULE_FOR_ELECTRONIC_TRANSFORMER_TYPE (0x07), il modulo è un VMB1DM 03 = il modulo è programmato per la funzione "dimmer con memoria" 0F = timer disabilitato 81 = versione firmware 07 = anno di produzione 02 = settimana di produzione

VMB1RY e VMB4RY sono simili in quanto azionano dei relé nello stesso modo; vale la stessa cosa per i controlli tapparella VMB1BL e VMB2BL e per i dispositivi di ingresso VMB6IN e VMB8PB (il dimmer VMB1DM attualmente è l'unico nel suo genere...).

Esaurite le premesse, siete pronti per iniziare ad interagire con il sistema seguendo i nostri esempi pratici: scaricate i manuali utente dal sito www.velleman.be perché da questo momento diventano indispensabili. Come ultima nota, gli esempi ri-

guardano l'impianto domestico descritto nei fascicoli precedenti.

Scansione della rete

Questa procedura permette di identificare i dispositivi presenti in rete e l'indirizzo assegnato ad essi. Si esegue inviando il comando "Module Type Request", il quale è riconosciuto da tutti i dispositivi: esso ha sempre priorità bassa (SID10 e SID9 a "11"), necessita dell'RTR a "1", non contiene databyte (DLC3÷0 = 0000) ed invia solamente il valore numerico dell'indirizzo che si sta interro-

gando. Il dispositivo che riconosce il proprio indirizzo nel comando "Module Type Request" risponde inviando il messaggio "Command Module Type".

In Tabella 1 trovate il log del traffico dati generato da tutti i dispositivi presenti in rete, in risposta alla scansione degli indirizzi 01 ÷ 07. Per facilitare la comprensione, abbiamo colorato in modo differente i vari byte: i delimitatori di stringa sono blu, l'indirizzo del dispositivo (mittente o destinatario) è rosso, i byte relativi all'RTR e ai DATA BYTE

sono viola, infine il checksum è verde. Nella colonna di destra abbiamo commentato dettagliatamente ogni byte transitato.

Attivazione del relé sul modulo VMB1RY

Identifichiamo sul manuale i comandi "Switch Relay On" e "Switch Relay Off". Entrambi hanno priorità alta (SID10 e SID9 = 00) e RTR a "0". Sono composti da due databyte (DLC3÷0 = 0010) in cui il primo identifica l'azione da compiere (0x02 = attiva relé, 0x01 = disattiva relé) ed il secondo il relé da controllare

Tabella 2

OF F8 01 02 02 01 F3 04	
OF	inizio stringa
F8	priorità alta
01	indirizzo destinatario del comando (il modulo VMB1RY)
02	due databyte da inviare
02	COMMAND_SWITCH_RELAY_ON comando di attivazione relé
01	attivazione del primo relé (in questo caso è uno solo)
F3	checksum
04	terminatore di stringa

(nel caso in esame, si ha un solo relé identificato da B"00000001", equivalente a 0x01). Per attivare il relé, il PC deve inviare il comando visibile in Tabella 2. Il modulo risponde inviando due stringhe differenti. Con la prima (Tabella 3) esso chiede al modulo VMB6IN di accendere il led del pulsante 1, pulsante usato per controllare il relé. Con la seconda, invece (Tabella 4), il VMB1RY informa tutta la rete di avere attivato il

Tabella 3

OF FB 04 02 F6 01 F9 04	
OF	inizio stringa
FB	priorità bassa
04	indirizzo destinatario del comando (il modulo pulsanti VMB6IN)
02	due databyte da inviare
F6	COMMAND_SET_LED (accensione dei led)
01	accende il led del pulsante 1 sul modulo VMB6IN
F9	checksum
04	terminatore di stringa

Tabella 4

OF FB 01 08 FB 01 07 01 80 00 00 00 69 04	
OF	inizio stringa
FB	priorità bassa
01	indirizzo mittente (modulo VMB1RY)
08	otto databyte trasmessi
FB	COMMAND_RELAY_STATUS (stato del relé)
01	identifica il relé numero 1 (unico in questo caso)
07	funzione di dual-timer (con tempo 0)
01	relé attivo
80	led di stato relé acceso
00	MSB timer
00	2° byte timer
00	LSB timer
69	checksum
04	terminatore di stringa

relé, il led di segnalazione di stato ed infine di non avere alcuna operazione di timer in corso.

Richiesta del nome al modulo VMB1DM

È l'ultimo esempio proposto, anche perché, dopo aver compreso la procedura, occorre solo fare un po' di pratica. Il comando per richiedere il nome è visibile in Tabella 5. Il

Tabella 5

OF FB 07 02 EF 01 FD 04	
OF	inizio stringa
FB	priorità bassa
07	indirizzo destinatario del comando (il modulo VMB1DM)
02	due databyte da inviare
EF	Comando di richiesta del nome
01	Richiesto il nome del primo canale (in questo caso è uno solo)
FD	checksum
04	terminatore di stringa

Tabella 6

OF FB 07 08 F0 01 44 69 6D 6D 65 72 98 04	
OF	inizio stringa
FB	priorità bassa
07	indirizzo mittente del messaggio (il modulo VMB1DM)
08	otto databyte da inviare
...	Risposta in testo: ~Dimmer (8 byte)
01	checksum
FD	checksum

Tabella 7

OF FB 07 08 F0 01 44 69 6D 6D 65 72 98 04	
---	--

dimmer risponde con quattro stringhe a bassa priorità: esse contengono i caratteri del nome. Nel nostro esempio, il nome è contenuto tutto nella prima stringa (Tabella 6), mentre le tre successive (simili a quella in Tabella 7) contengono solo caratteri fittizi.

Conclusioni

Terminiamo qui la presentazione dei prodotti Velbus e del relativo protocollo. Come di consueto, però, la mera teoria non ci basta: continuate a seguirci perché pubblicheremo a breve un progetto pratico per attivare qualsiasi dispositivo Velbus tramite un telecomando a 2 o 4 canali.

per il

MATERIALE

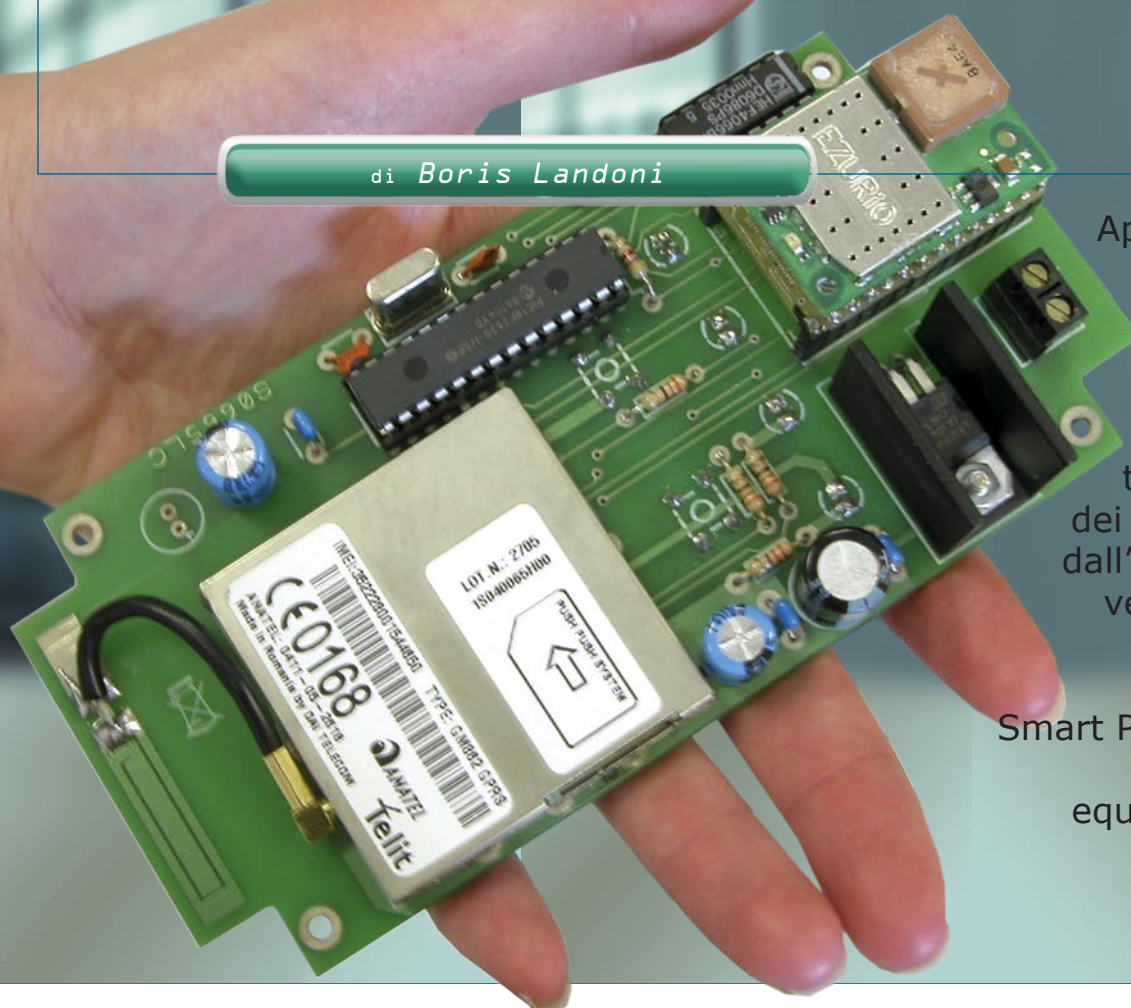
L'applicativo VelbusLink Versione 5.1 e la libreria *velleman.velbus.dll* sono scaricabili dal sito www.velleman.be mentre il programma EeasyVelbus è scaricabile gratuitamente dal sito della rivista (www.elettronica.in.it). Le interfacce Velbus per PC cod. VMB1RS e cod. VMB1USB costano rispettivamente 44,00 e 53,00 Euro e come tutti gli altri moduli Velbus sono distribuite da Futura Elettronica. I prezzi si intendono IVA compresa.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-778112 ~ <http://www.futuranet.it>

Bridge GSM Bluetooth

di Boris Landoni



Appositamente pensato per i localizzatori presentati in passato, effettua il trasferimento dei dati in arrivo dall'unità remota verso Personal Computer, PC palmari, Smart Phone ed altri dispositivi equipaggiati con interfaccia Bluetooth.

Tra i tanti progetti di dispositivi di controllo remoto che abbiamo pubblicato, uno dei più apprezzati è sicuramente il localizzatore GPS/GSM descritto nel fascicolo n° 100 (cod. FT596), un dispositivo miniaturizzato, alimentato a batteria e provvisto di modulo GSM tramite il quale può sia essere interrogato a distanza, al fine di conoscere la posizione del veicolo nel quale è installato, sia inviare esso stesso SMS o e-mail tramite il servizio SMS con cadenza prefissata o in caso si verifichi qualche specifico evento (messa in moto della vettura, attivazione del sensore di movimento, ecc). Questo

localizzatore è stato sviluppato essenzialmente per la localizzazione di autoveicoli e per seguirne il percorso (gestione delle "flotte aziendali"). Il grande interesse suscitato dal progetto ci ha spinti, poco tempo dopo, a pubblicarne una versione nella quale è stata implementata anche la modalità di tracking in real-time: chiamando il dispositivo, questi risponde in modalità dati e comunica in tempo reale le informazioni sul posizionamento, estraendo dalle stringhe NMEA0183 le sole coordinate. In considerazione del fatto che tutti i localizzatori remoti utilizzano per l'invio dei dati la rete GSM e che



nei computer (desktop, portatili, palmari, Smart Phone e PDA) si usa sempre più l'interfaccia Bluetooth, abbiamo pensato di progettare e proporre un "adattatore" capace di catturare le informazioni ricevute via cellulare e trasferirle, appunto, via Bluetooth ad altri dispositivi. Così è nato il progetto qui descritto, il quale, ad onor del vero, più che una semplice interfaccia può essere con-

siderato effettivamente un bridge, ossia un ponte di collegamento fra la rete telefonica radiomobile e un sistema Bluetooth, perlomeno per quanto che riguarda i dati provenienti dai sistemi di localizzazione.

Come funziona

Il dispositivo che proponiamo è equipaggiato con un modulo GSM e GPRS al quale giungono gli SMS

trasmessi dai localizzatori FT596 preventivamente programmati per utilizzare la modalità GPRMC. Attenzione: se non impostate tale modalità, il bridge non può funzionare. Per meglio comprendere il funzionamento del bridge bisogna avere ben chiaro il modo di comunicare dei nostri localizzatori: la prima versione (quella proposta nel fascicolo n° 100) invia le coordinate utilizzando ➤

Modalità Polling



Fig. 1

SMS in automatico

Il Bridge rimane in attesa di un SMS da un localizzatore noto. Ricevuto l'SMS e verificato che si tratta di una sentenza corretta, i dati vengono elaborati ed inviati alla porta Bluetooth.

SMS o inviando e-mail tramite il servizio SMS e lo fa su sollecitazione esterna o periodicamente in modo autonomo; nel primo caso (modalità fonìa) attende di ricevere una telefonata, alla quale non risponde, poi riaggancia e prepara un messaggio di testo contenente le coordinate o il contenuto dell'eventuale e-mail, che spedisce al numero di telefono dal quale ha ricevuto la chiamata. Nella modalità periodica, che si attiva inviando un apposito SMS di configurazione, il localizzatore invia un SMS con le coordinate del momento (o il testo delle e-mail corrispondenti) al numero telefonico definito nello stesso SMS di configurazione. Nella versione descritta nel fascicolo n° 111, il localizzatore portatile svolge ancora le funzioni appena descritte, ma aggiunge quella in real-time: quando viene chiamato, risponde (modalità dati) e comunica le coordinate così come le riceve dalla sua sezione GPS; per ragioni di sicurezza, accetta la comunicazione solo se la chiamata proviene da un numero preventivamente comunicatogli in fase di configurazione.

Il bridge descritto in queste pagine supporta tutte le modalità previste dal localizzatore FT596: può chiamarne uno o più (non contemporaneamente, s'intende...) sia per chiedere l'invio in real time dei dati di posizionamento, sia per forzare la trasmissione, da parte del localizzatore

chiamato, di SMS contenenti sempre le informazioni sulla posizione in cui si trova l'FT596 del caso. Per attivare le funzioni di chiamata bisogna agire manualmente, premendo appositi pulsanti dei quali è dotato e che descriveremo tra breve.

Quanto detto riguarda il funzionamento attivo; ma il nostro bridge può anche restare passivo e attendere che uno o più localizzatori configurati per la trasmissione ciclica, nei quali è memorizzato il numero della SIM card inserita nel suo modulo cellulare, gli mandino gli SMS con le coordinate.

Alla luce di ciò, possiamo analizzare una ad una le modalità di funzionamento, implementate per interfacciarci correttamente ai localizzatori;

lo facciamo con la premessa che il dispositivo, prima dell'utilizzo, deve essere configurato (le operazioni del caso si conducono mediante SMS) al fine di memorizzare i numeri degli eventuali localizzatori che deve chiamare e dai quali può accettare ed elaborare i messaggi in arrivo. Altrimenti, se riceve un SMS da un numero sconosciuto lo rifiuta.

Iniziamo l'esame con la modalità Polling (Fig. 1), nella quale il circuito, prima di compiere qualsiasi operazione di comunicazione, attende un messaggio di testo in arrivo da uno dei localizzatori autorizzati, cioè da uno di quelli il cui numero sia stato preventivamente memorizzato con l'apposita procedura. Alla sua ricezione, verificata la provenienza e accertato che il numero del mittente sia in lista, estrae il contenuto ed elabora i dati GPRMC, ovvero le coordinate; a questo punto, prepara la stringa corrispondente e la ritrasmette mediante la sezione Bluetooth, verso gli apparati che possono riceverla. La modalità polling viene di solito utilizzata quando con un computer bisogna monitorare posizioni e percorsi di uno o più localizzatori, realizzando un logger: i record corrispondenti ad ogni FT596 vengono scritti in ordine

Modalità On Demand



Fig. 2

SMS in automatico

Premendo il pulsante P2, il Bridge esegue una chiamata voce al localizzatore. Questi non accetta la chiamata, ma verifica il numero del mittente: se il mittente corrisponde al bridge, il localizzatore invia al nostro circuito un SMS in risposta contenente le coordinate.

Chiamata Vocale

cronologico in un apposito file, così il personale addetto può, all'occorrenza, richiamarli e visualizzare il tracciato nello schermo del PC, utilizzando un adeguato programma di navigazione e la cartografia del caso.

Quando si desidera un'informazione immediata sulla posizione o sul percorso di un certo localizzatore, bisogna ricorrere alle modalità che prevedono l'interrogazione manuale, che esaminiamo prendendo in considerazione per prima quella corrispondente all'invio in tempo reale (Fig. 3). Si avvia premendo il pulsante P1 e l'esecuzione in corso viene confermata dall'accensione del led LD1 (verde); subito viene composto il numero corrispondente alla prima posizione della lista di quelli abilitati (otto in tutto) e, alla risposta del localizzatore, il circuito fa illuminare il led LD3 (giallo) che resta acceso per tutta la durata della comunicazione fra FT596 e bridge. A riguardo, va detto che la telefonata può essere conclusa solamente dall'utente, premendo una seconda volta P2; certo, può interrompersi anche per problemi tecnici, quali la scarica della batteria del localizzatore, la disconnessione dalla rete GSM, la perdita del segnale del sistema GPS, ma questo è un altro discorso. Comunque, in caso di perdita della comunicazione con il localizzatore il bridge interrompe la telefonata. L'interruzione manuale o forzata viene segnalata dall'accensione per 4 secondi del led LD2 (rosso).

Durante il colloquio (led giallo acceso) il nostro dispositivo legge i dati GPRMC e li trasferisce, opportunamente elaborati, al modulo Bluetooth, il quale, dopo aver negoziato la connessione, li trasmette verso l'apparecchio (Personal Computer o altro) nel suo campo di copertura, con il quale ha instaurato la comunicazione; per fare in modo che le informazioni sulle coordinate vengano lette anche da apparati

standard, quali i PC palmari dotati di software di navigazione satellitare, il bridge ricostruisce delle finte stringhe NMEA0183, completando le coordinate con dati fittizi, seppure verosimili e nel formato previsto e riconosciuto, per esempio, dal famoso Tom-Tom. Insomma, aggiunge alle GPRMC i dati su numero dei satelliti, livello dei segnali ecc., generati localmente, anche se non reali.

Bene, spiegato come avviene la telefonata in modalità dati (real-time), possiamo andare a vedere come

localizzatore prepara un SMS contenente le coordinate del momento e lo invia proprio ad esso; il nostro dispositivo lo riceve, lo elabora e prepara una stringa standard NMEA0183, secondo le stesse modalità descritte poc'anzi per il bridging GSM/Bluetooth in tempo reale. A questo punto, l'interfaccia Bluetooth cerca la connessione con l'apparato destinatario delle coordinate e, stabilita la comunicazione, trasmette la stringa, ripetutamente, per cinquanta volte di fila. Fatto ciò, il bridge torna auto-

Modalità Real Time



Fig. 3



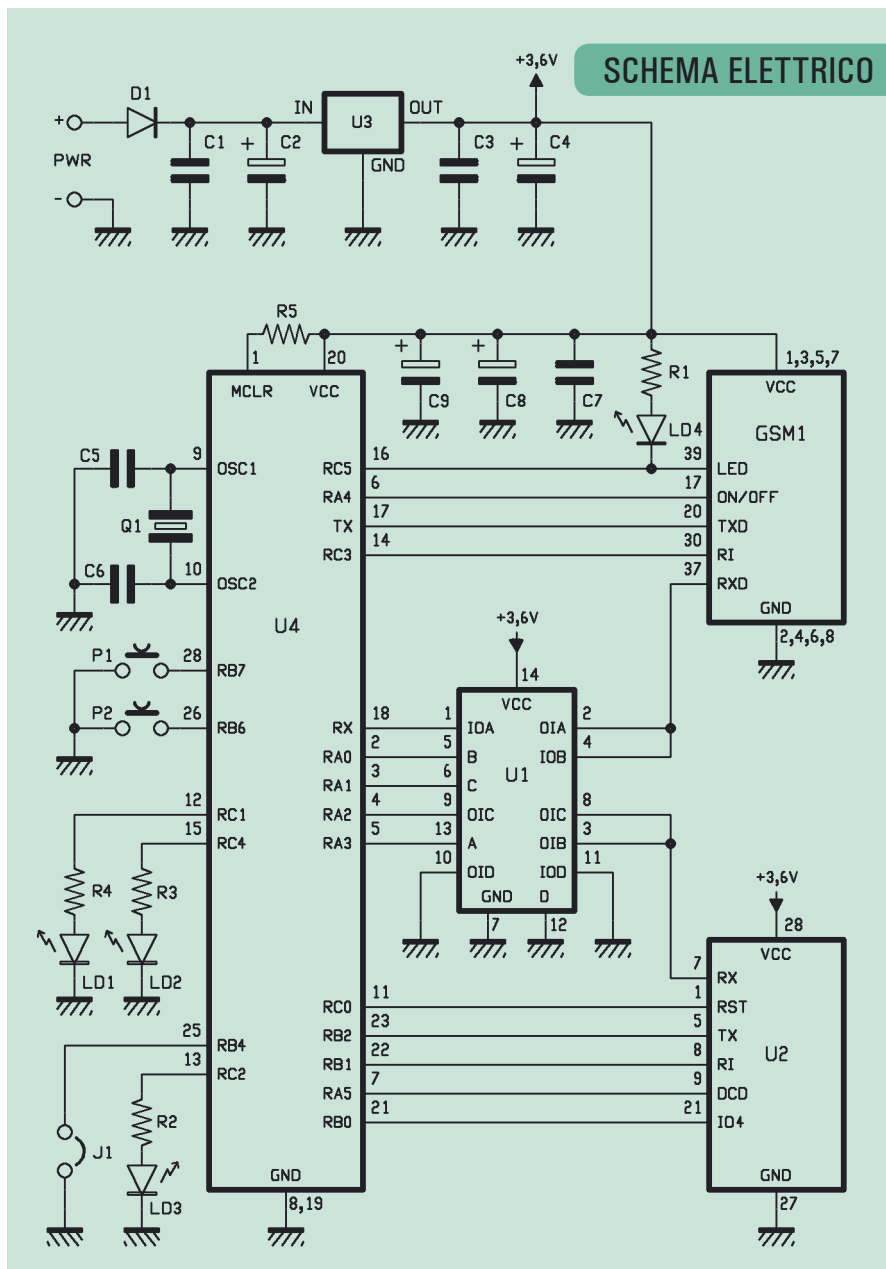
Premendo il pulsante P1, il Bridge esegue una chiamata dati al localizzatore. Questi verifica il numero del mittente e, riconosciuto il bridge, accetta la chiamata aprendo un canale dati sul quale invia le coordinate in modo continuo.

funziona quella che prevede la richiesta di SMS (Fig. 2). Si attiva, al solito, premendo un pulsante, che però, stavolta, non è P1 ma P2; ciò fa partire una telefonata (segnalata ancora dall'illuminazione dell'LD1) diretta sempre al primo dei numeri telefonici memorizzati nella lista di quelli abilitati. Quando la riceve, il localizzatore corrispondente non risponde ma "butta giù" la chiamata, determinando l'invio del tono di occupato e della segnalazione di rifiuto della chiamata lungo la rete telefonica radiomobile; il nostro bridge manifesta tale condizione accendendo il led rosso. Intanto, se riconosce il numero del bridge (deve essergli stato preventivamente memorizzato...) il

maticamente a riposo e attende un nuovo evento (pressione di uno dei pulsanti o ricezione di un SMS).

Schema elettrico

Il circuito che realizza le funzioni descritte non può che essere complesso, ma nonostante ciò risulta abbastanza compatto facendo ricorso a specifici moduli che assolvono ognuno a precisi compiti: il cellulare GSM/GPRS della Telit provvede all'invio ed alla ricezione degli SMS, sia di configurazione che di dialogo con i localizzatori, oltre che all'effettuazione delle chiamate di richiesta delle coordinate, dirette sempre ai localizzatori; il modulo Ezurio, che gestisce il colloquio lungo il canale ➤



radio Bluetooth e provvede quindi a passare a un computer, pocket PC, PDA o altro le informazioni sulle coordinate.

Il tutto viene gestito da un microcontrollore PIC18F2620 di Microchip, programmato con un firmware che, dopo l'accensione, per prima cosa inizializza le linee RC1, RC2 ed RC4 come uscite per il comando dei led di segnalazione dell'attività della sezione telefonica, RA4 ancora da uscita per gestire accensione e spegnimento del modulo Telit, quindi RC3 ed RC5 come input, destinati rispettivamente alla

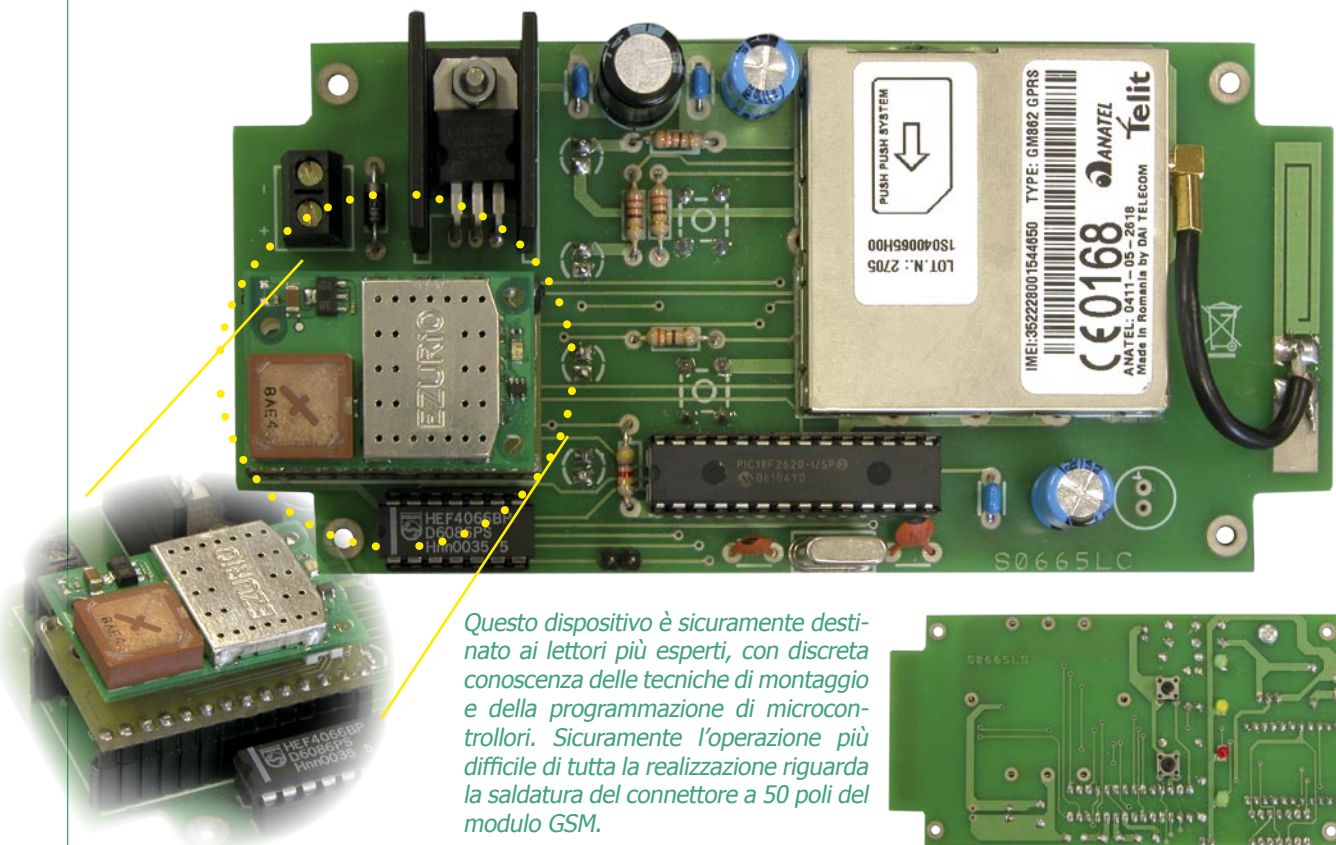
lettura dell'uscita per il led di stato della connessione GSM e di quella che segnala l'arrivo di una chiamata; stiamo parlando ancora del modulo cellulare. A riguardo va detto che il cellulare Telit usa RI (Ring Indicator, normalmente ad 1 logico) per comunicare quando sta ricevendo una telefonata; LED serve invece a far accendere il led di stato, che pulsa con cadenza 1/1 sec. quando il GM862 è in ricerca del segnale della rete GSM ed emette un impulso a zero logico da 0,5 secondi seguito da una pausa (allo stato alto) di 2 secondi quando, invece, è stata stabili-

ta la connessione. Questo per quanto riguarda la gestione del cellulare.

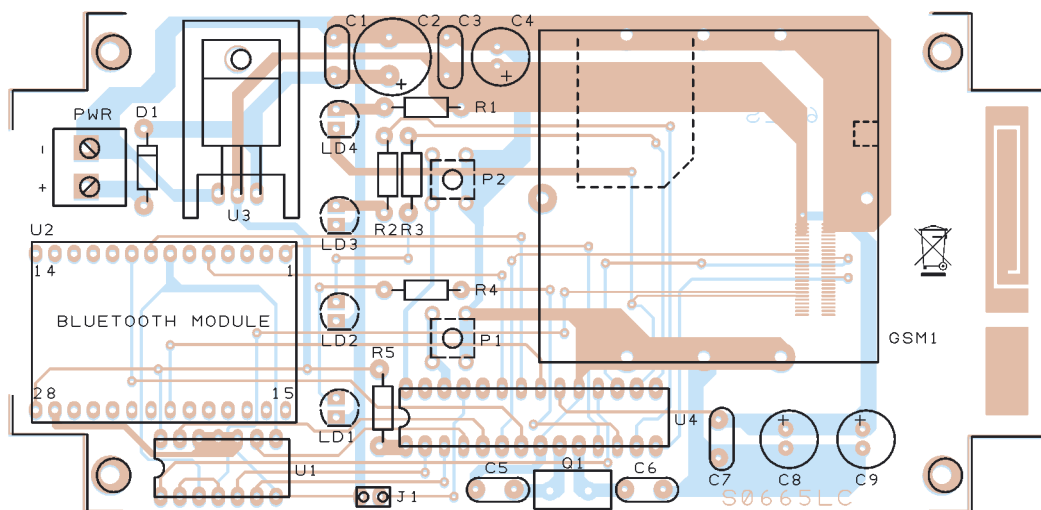
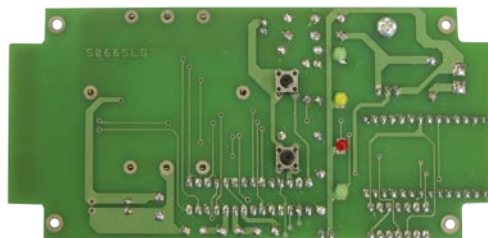
Sempre all'inizializzazione, vengono definite le linee di controllo del modulo BISM2 Ezurio: RC0 diventa l'uscita usata per resettare il componente, RB0 un input, sfruttato per verificare la presenza di segnale radio Bluetooth mediante la lettura dell'output IO4 destinata, nel BISM2, al led di stato; RB2 è ancora un input, che il software del PIC usa per leggere i dati in arrivo dal link Bluetooth quando viene negoziata una sessione di comunicazione con un altro dispositivo (PC, PDA e simili). Infine RB1 ed RA5 vengono configurate sempre come ingressi, per leggere, rispettivamente, la condizione della linea RI (che si attiva quando un dispositivo Bluetooth tenta di comunicare con il nostro modulo) e della DCD (Data Carrier Detect della seriale del modulo) dell'Ezurio.

Un altro set di I/O viene inizializzato per commutare i canali RXD del modulo GSM ed RX del BISM2, pilotando opportunamente una matrice composta da interruttori CMOS, sfruttata per far dialogare i moduli ora con il microcontrollore (mediante canali dati distinti) ora fra loro; le linee del caso sono tre, tutte impostate come uscite. RA0 controlla lo switch B, ossia quello che porta l'RXD, del Telit, che è un'uscita, sull'RX del modulo Bluetooth; RA1 gestisce lo switch C, che fa dialogare RA2, inizializzata come linea seriale dei dati in uscita dal micro, con l'U2; infine, a RA3 è affidato lo switch A, che serve a far ricevere al microcontrollore i dati in arrivo dal modulo GSM/GPRS.

All'inizializzazione viene anche attivato l'UART interno al PIC18F2620, che usiamo per dialogare con il cellulare Telit; a proposito, si noti che le linee TX ed RX sono, rispettivamente, uscita e ingresso dati. Le corrispondenti del GM862 sono TXD ed RXD; ciò ➤



Questo dispositivo è sicuramente destinato ai lettori più esperti, con discreta conoscenza delle tecniche di montaggio e della programmazione di microcontrollori. Sicuramente l'operazione più difficile di tutta la realizzazione riguarda la saldatura del connettore a 50 poli del modulo GSM.



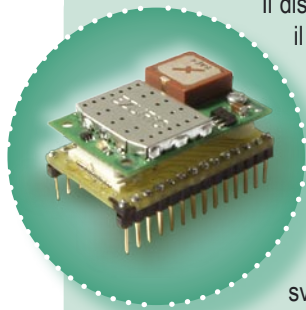
ELENCO COMPONENTI:

R1÷R4: 330 ohm
 R5: 4,7 kohm
 C1, C3, C7: 100 nF multistrato
 C2: 470 µF 25 VL elettrolitico
 C4: 470 µF 16 VL elettrolitico
 C5, C6: 10 pF ceramico
 C8: 470 µF 16 VL elettrolitico
 C9: 470 µF 16 VL elettrolitico
 U1: CD4066
 U2: Modulo interfaccia BISM2 (FT622M)
 U3: LD1086-3.6

U4: PIC18F2620 (MF665)
 GSM1: GM862
 Q1: Quarzo 20 MHz
 D1: 1N4007
 LD1, LD4: led 3 mm verde
 LD2: led 3 mm rosso
 LD3: led 3 mm giallo
 P1, P2: Microswitch

Varie:
 - Morsettiera 2 poli

- Zoccolo 7+7
 - Zoccolo 14+14
 - Connettore 50 poli per GM862
 - Strip femmina 14 pin (2 pz.)
 - Strip maschio 2 poli
 - Jumper
 - Dissipatore (ML26)
 - Vite 10 mm 3 MA
 - Dado 3 MA
 - Cavo antenna GSM con attacco MMCX
 - Circuito stampato codice S665



Il dispositivo Bluetooth che abbiamo usato in questo progetto è il noto BISM2, un completo transceiver in Classe 1, il cui trasmettitore ha una potenza che può variare tra 0 e +6 dBm e, in assenza di ostacoli, consente una portata teorica di 250 metri. Opera tra 2.400 e 2.485 MHz, con una sensibilità in ricezione di ben -90 dB (il guadagno dell'antenna ceramica integrata è di +2 dB). Il data-rate raggiunge i 300 kbps ed è più che sufficiente a trasportare discrete quantità di dati in tempi decisamente ridotti. Il modulo, prodotto dalla Ezurio (www.ezurio.com) dispone di una porta seriale a livello TTL che, mediante un convertitore RS232, permette di connetterlo facilmente ad un PC di cui può diventare, a tutti gli effetti, una periferica. Leggero e compatto, il modulo misura appena 22,8x33,8 mm. Il punto di forza di BISM2 (BISM significa "Bluetooth Intelligent Serial Module") è costituito da un set di comandi AT molto ben strutturato: tanto completo quanto intuitivo, tale set di comandi consente agli sviluppatori di integrare facilmente l'interfaccia Bluetooth in sistemi a microprocessore senza la necessità di dover conoscere tutto il protocollo H4 (HCI), protocollo, questo, progettato dal SIG (Special Interest Group).

non deve apparire insensato perché il modem del modulo Telit ha le linee siglate come vuole la convenzione: trattandosi di un DCE (Data Communication Equipment) TXD è l'ingresso dei dati in arrivo dal TX del terminale (in questo caso il PIC) ed RXD l'uscita diretta ad RX dello stesso terminale. Quanto alla comunicazione seriale con il BISM2, dato che l'UART è già impegnato, viene effettuata mediante una porta seriale costruita usando RA2 (trasmissione) ed RB2 (ricezione).

Per concludere il discorso sulla fase di inizializzazione, diciamo che le linee RB6 ed RB7 vengono impostate come ingressi con pull-up interno per leggere lo stato dei pulsanti; analogo è il discorso per la linea collegata al jumper (RB4) per il momento non usato e riservato a future applicazioni.

Completata l'inizializzazione, gira il main program del micro, che prevede il test ciclico sia dello stato di P1 e P2, sia dell'uscita per il led di stato del modulo GSM1; il controllo è mirato -nel primo caso- a riconoscere la richiesta da parte dell'utente di chiamata verso un localizzatore, e nel secondo a rilevare l'arrivo di SMS, che solitamente determina il rapido lampeggio (0,5/0,5 sec) del predetto diodo luminoso (LD4).

Quando è l'utente a richiedere manualmente l'uso del bridge GSM/Bluetooth, il micro lancia due routine differenti a seconda che ad essere premuto sia stato P1 o P2: nel primo caso, inizialmente attiva lo switch

CMOS A e interroga il cellulare per accertare che il localizzatore sia connesso, quindi, non appena quest'ultimo sta per trasmettere i dati sulle coordinate in tempo reale, disabilita lo switch C e attiva il B; contemporaneamente sconnette lo switch C, fino a quel momento utilizzato per controllare la presenza di segnale Bluetooth e negoziare la connessione con il BISM2. In tal modo, realizza un ponte fra il canale dati di uscita del GM862 (RXD) e quello d'ingresso (RX) del modulo Ezurio, consentendo l'invio in real-time delle informazioni ricevute dalla rete radiomobile, direttamente sul canale Bluetooth. Il ponte e quindi lo switch CMOS B vengono interrotti quando la comunicazione telefonica dati viene sospesa, per intervento dell'operatore o per causa esterna. A questo punto riprende a girare il main-program.

Vediamo adesso cosa avviene quando si preme P2 e si inizia una sessione di lavoro che prevede la chiamata e l'attesa di un SMS dal localizzatore: il microcontrollore chiude inizialmente lo switch A e impartisce al cellulare le istruzioni per l'effettuazione della telefonata, poi attende che l'FT596 rifiuti la chiamata, cosa che accerta dai dati che il GM862 invia lungo l'RXD; a questo punto si mette in attesa dell'SMS contenente le coordinate e, quando lo riceve, elabora i dati corrispondenti, costruisce la stringa NMEA e la fa passare dallo switch C, che nel frattempo ha chiuso, af-

finché raggiungano l'RX del modulo BISM2 e vengano trasmessi sul canale Bluetooth. Esaurita la trasmissione, il PIC disattiva lo switch C e riprende a girare il main-program.

Viste le due modalità manuali, non resta che analizzare quella automatica (passiva) corrispondente alla ricezione di un SMS da un localizzatore (senza la preventiva chiamata) che lavora in modalità di invio ciclico delle coordinate: normalmente, in assenza di comando manuale, il microcontrollore chiude lo switch A e attende dal cellulare la comunicazione di SMS ricevuto; quando riceve il messaggio di testo, elabora i dati corrispondenti, costruisce la stringa NMEA e la fa passare dallo switch C, che nel frattempo ha chiuso, affinché raggiungano l'RX del modulo BISM2 e possano essere trasmessi via Bluetooth. Poi torna a riposo e disattiva lo switch C. Riprende a girare il main-program.

Completiamo l'analisi dello schema elettrico con la sezione di alimentazione: il circuito funziona con 8÷20 V in continua, da applicare ai punti + e - PWR con polarità positiva sull'anodo del D5; qualora l'alimentazione venga applicata con polarità invertita, tale diodo protegge il resto della componentistica perché in tal caso non conduce. I condensatori C1 e C2 filtrano la componente continua da disturbi a radiofrequenza ed eventuale residuo d'alternata; ai loro capi viene prelevata la tensione che il regolatore integrato LD1086-3.6 riduce a 3,6 V esatti e ben sta-

Tabella 1

Funzione	Comando SMS	Risposta SMS	PWD obbligatoria
Cambio Password	PWDxxxx;12345	La nuova password è stata memorizzata	Sì
Memorizzare un numero (max 8 numeri, max 19 caratteri per numero)	NUMx +393355760937; 12345	Numeri memorizzati: Pos1 + nnnnnnnnn; Pos2 + nnnnnnnnn; ...; Pos8 + nnnnnnnnn	Solo se la posizione "x" non è vuota e se il numero del telefono che invia l'SMS non è in lista
Cancellare un numero	NUMx;12345	Come sopra	Sì
Verificare i numeri memorizzati	NUM?;12345	Come sopra	Sì
Reset completo	RES;12345	Reset Eseguito	Sì
Nota: la password predefinita è "12345" ed è localizzata negli indirizzi di memoria 1001÷1005.			

bilizzati. Il componente è un integrato a tre piedini che può erogare una corrente più che sufficiente ad alimentare il modulo Bluetooth, il microcontrollore e il cellulare GSM anche nelle condizioni più gravose (telefonata in corso e trasmissione da parte del BISM2).

Realizzazione e collaudo

Sul piano costruttivo, il bridge non presenta particolari difficoltà, almeno per lettori esperti cui questo progetto è, ovviamente, destinato; l'elemento più critico, cioè l'antenna del modulo GSM/GPRS, è già realizzata da una pista del circuito stampato, quindi basta preparare la basetta ricorrendo alla tecnica di fotoincisione seguendo fedelmente le tracce lato rame (si trovano sul nostro sito Internet www.elettronica.in) così non ci sarà alcun problema. Tutte le saldature dovranno essere effettuate con un saldatore da 20÷30 W di potenza a punta molto sottile, usando filo di lega saldante da 0,5 mm di diametro e del flussante. Il primo componente da montare è il connettore miniatura per il modulo GSM/GPRS, che suggeriamo di posizionare ben centrato sulle piazzole corrispondenti e di fermare stagnando leggermente un elettrodo per lato; si raccomanda di prestare molta attenzione a non cortocircuitare, con lo stagno, contatti attigui che non devono essere uniti.

Sistemato il connettore, si possono inserire e saldare i componenti tradizionali, andando in ordine di

altezza (resistenze, diodi al silicio, pulsanti da c.s. e zoccoli, poi led e condensatori), prestando la dovuta attenzione al verso degli elementi polarizzati. Per il microcontrollore e il 4066 utilizzate degli zoccoli dip; quanto al montaggio del modulo FT622M contenente il BISM2, conviene procurarsi uno zoccolo da integrato a 28 piedini, quindi tagliarlo longitudinalmente fino a ricavare due strip da 14 pin l'una, che salderete nelle file di fori corrispondenti (BLUETOOTH MODULE).

Il regolatore integrato va montato sdraiato su un dissipatore di calore sagomato ad "U", da 15÷18 °C/W, che va disposto per primo; quindi si piegano i tre terminali dell'U3 ad angolo retto, li si infila nei rispettivi fori, poi si fissa il componente con un piccolo bullone da 3MA (che blocchi anche il dissipatore allo stampato) e si procede alla stagnatura delle piazzole corrispondenti. Sistemato il tutto, bisogna inserire a fondo il cellulare GM862, mantenendolo ben parallelo alla superficie del circuito stampato e stagnando le linguette di fissaggio, così da bloccarlo.

L'antenna integrata va collegata al modulo Telit mediante un cavetto coassiale terminante da un lato con l'apposito connettore MMCX; l'altro capo del cavo va sguainato, quindi la maglia di schermo va raccolta attorno alla guaina e tagliata per evitare che tocchi qualche pista. L'anima del coassiale va stagnata sul punto più esterno della pista d'an-

tenna. L'eventuale contenitore in cui verrà inserito il dispositivo dovrà essere in plastica e comunque non metallico, altrimenti farebbe da schermo elettromagnetico ed ostacolerebbe le onde radio, impedendo al cellulare Telit di comunicare. Si può usare un involucro metallico, ma bisogna portare l'antenna

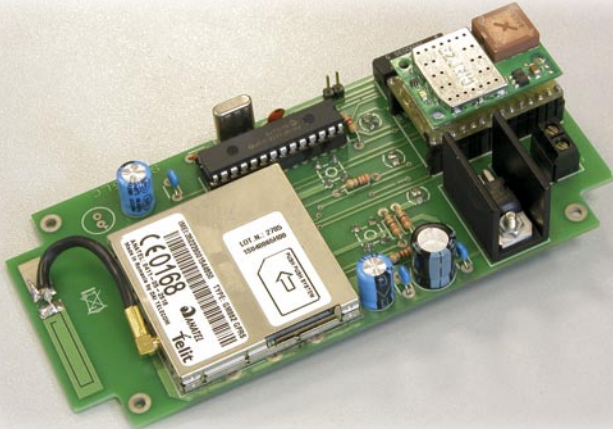
all'esterno, usando un cavetto adattatore provvisto di connettore FME da fissare all'esterno di una parete. In ogni caso, il contenitore va forato per rendere visibili i quattro diodi luminosi, ma anche per far uscire i fili di collegamento alla morsettiera di alimentazione.

Per far funzionare il bridge occorre un alimentatore capace di erogare da 8 a 20 volt e una corrente continua di almeno 600 mA.

Le procedure di programmazione

Prima di utilizzare il bridge GSM/Bluetooth bisogna effettuare un minimo di programmazione, che serve a configurarlo perché sappia da chi accettare gli SMS e a quali numeri effettuare le chiamate. Le operazioni del caso si svolgono mediante messaggi di testo inviati da un telefono cellulare; allo scopo di impedire accessi non autorizzati, la programmazione è protetta da password, la quale, per impostazione predefinita (o dopo un reset completo) è 12345. Per modificarla, occorre inviare un SMS che deve contenere il testo: PWDxxxx;attpwd, nel quale al posto delle x si devono scrivere le cinque cifre della password che devono prendere il posto di quella attuale (attpwd) che va inserita per completare il comando e dargli effetto. Se è la prima volta che si tenta la modifica o è stato eseguito un reset completo, attpwd è 12345; al comando il circuito risponde inviando al numero telefonico che ha impartito il co-





mando un SMS del tipo: *La nuova password è stata memorizzata*. La memorizzazione di un numero della lista degli otto che il microcontrollore del sistema può memorizzare si esegue inviando un SMS che contenga il seguente testo (senza spazi): `NUMx+nnnnnnnnnn;attpwd`; in esso `nnnnnnnnnn` sta per il numero di telefono, compreso il prefisso nazionale (per esempio `+393398888888`) e `attpwd` è la password attualmente valida. Al posto della `x` si deve scrivere il numero (1÷8) corrispondente alla posizione in cui si desidera memorizzare l'indicativo telefonico. Si rammenti che il numero non deve, nel complesso, superare le 19 cifre; si ricordi altresì che tentando di scrivere un numero in una posizione nella quale ne è già memorizzato un altro, l'operazione ha l'effetto di rimuovere il vecchio numero e mettere al suo posto quello nuovo: notate che il comando di memorizzazione, se proveniente da uno degli otto numeri noti, deve essere provvisto di password quando la posizione su cui si vuole scrivere il nuovo numero è già occu-

pata. Se, invece, è vuota, la password può essere omessa. Ad ogni comando di memorizzazione o sovrascrittura, il telecomando risponde con un SMS contenente il testo: *Numeri memorizzati: Posx+nnnnnnnnnn*; in esso `x` è la posizione in cui è stato memorizzato il numero `nnnnnnnnnn`. Si rammenti altresì che nelle modalità manuali il sistema chiama sempre e solo il numero telefonico memorizzato in prima posizione; quindi se si intende fargli telefonare a un determinato localizzatore, in posizione 1 va salvato il numero della SIM card di quest'ultimo. Ora, qualcuno si sarà già domandato il senso di poter memorizzare otto indicativi telefonici se poi il dispositivo può chiamarne uno solo; ebbene, i numeri in lista definiscono i localizzatori da cui è possibile accettare gli SMS con le coordinate, nella modalità passiva e poterne memorizzare più d'uno permette di monitorare lo stato di più FT596 configurati per l'invio periodico degli SMS. Poco importa se il monitoraggio si limita a questo. Per rimuovere un numero memorizzato nella lista, si utilizza il comando SMS: `NUMx;attpwd`; in esso `x` è il numero della posizione da cancellare e `attpwd` la solita password attualmente valida. Per esempio, volendo cancellare dalla lista il numero telefonico memorizzato nella terza posizione, dopo aver

inviato un comando di reset totale o senza aver mai sostituito la password originaria, occorre inviare un SMS contenente il testo `NUM3;12345`. All'occorrenza, è possibile richiedere al bridge la lista dei numeri attualmente memorizzati; il comando da utilizzare è: `NUM?;attpwd`. Trattandosi di una richiesta, il telecomando risponde inviando al numero da cui è giunto il messaggio un SMS che elenca, uno di seguito all'altro, i numeri registrati, ciascuno correlato con la posizione che ha in memoria; il testo corrispondente è del tipo: *Numeri memorizzati: Pos1+nnnnnnnnnn, Pos2+nnnnnnnnnn ecc.* Chiudiamo la panoramica sui comandi, con quello che si dà per ripristinare nel sistema le impostazioni di fabbrica (reset totale): si tratta di un SMS contenente il testo: `RES;attpwd`. Anche in questo caso è prevista una risposta, del tipo: *Reset eseguito*.

Utilizzo del bridge

Per usare correttamente il dispositivo bisogna innanzitutto disporre di un apparato Bluetooth che possa ricevere i dati del caso: un PC con installato un software di navigazione satellitare e cartografia, ovvero un PDA, un palmare o altro simile. Per prima cosa bisogna accendere l'apparato, fargli iniziare la ricerca di dispositivi BT e stabilire la connessione (se si tratta di un PC con sistema operativo Microsoft bisogna assegnare una COM virtuale ed aprirla); a questo punto il bridge è pronto per lavorare.

per il

MATERIALE

Il progetto descritto in queste pagine impiega componenti facilmente reperibili. Dal sito della rivista (www.elettronica.in.it) è possibile scaricare il master della scheda madre e il codice sorgente del microcontrollore. Quest'ultimo è anche disponibile già programmato (cod. MF665, 18,00 Euro). Il modulo Bism2 completo di scheda di adattamento (cod. FT622M) costa 75,00 Euro mentre il modulo GM862 costa 120,00 Euro. I prezzi sono comprensivi di IVA.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-778112 ~ <http://www.futuranet.it>

Localizzatore Mambo, funzioni e programmazione

di Davide Scullino

Si chiama Mambo ed è stato sviluppato per trovare impiego a 360 gradi nell'ambito della localizzazione e del controllo a distanza di persone e automezzi; le sue funzioni e il nutrito set di istruzioni, permettono di sfruttarlo per svolgere tanti altri compiti di contorno, quali l'abbinamento a un PC palmare per realizzare un navigatore satellitare. Seconda puntata.



Nel fascicolo precedente abbiamo introdotto un dispositivo estremamente versatile, nato per le applicazioni inerenti alla localizzazione e personalizzabile per adattarlo al compito cui si intende destinarlo; si tratta di Mambo, localizzatore GPS (prodotto dalla Falcom) provvisto di cellulare integrato GSM/GPRS, del quale possono essere utilizzate tutte le funzioni vocali (per effettuare una normale conversazione o trasmettere l'audio rilevato nell'ambiente). Mambo dispone di una batteria ricaricabile che gli permette di funzionare autonomamente; può essere chiamato e quindi interrogato

a distanza da una postazione di controllo, ma non solo: può inviare SMS contenenti la posizione o effettuare esso stesso chiamate dirette a chi deve tenere sotto controllo i movimenti di una persona o di un veicolo. Ancora, dispone di un efficace sensore di movimento che può essere abbinato ad alcune delle sue funzioni, per esempio facendo accendere il dispositivo solo quando è in movimento. Sul coperchio sono presenti tre led bicolore e quattro pulsanti, ciascuno dei quali può svolgere compiti dipendenti dalla programmazione.

In tema di programmazione, va detto che Mambo è prov- ➤

visto di un'interfaccia Bluetooth, la quale permette sia di dialogare con il modulo durante le fasi di impostazione e programmazione, sia di fare tutto ciò che un cellulare fa con il Bluetooth.

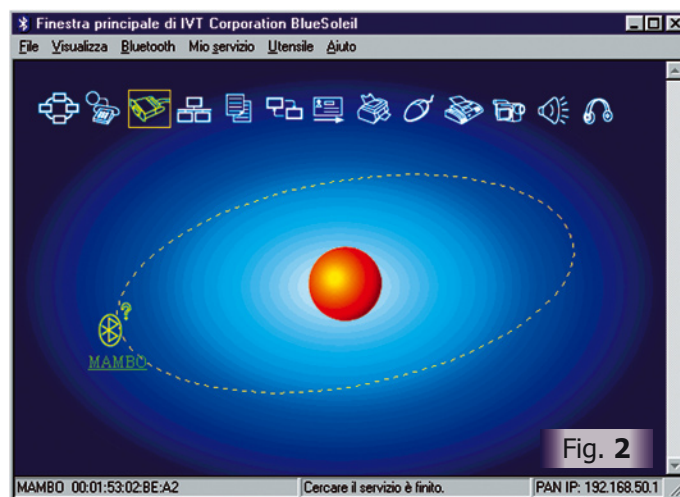
Mambo è governato da un microprocessore RISC che esegue i pochi compiti necessari a dialogare con il computer per caricare il programma di funzionamento prescelto; nella sua Flash EPROM è dunque collocato quello che possiamo considerare il suo BIOS, l'insieme delle istruzioni fondamentali per attivare una connessione Bluetooth.

Spiegate le funzioni di massima, questo mese entriamo nel vivo delle operazioni di programmazione, vedendo come si instaura la comunicazione via Bluetooth, come si interroga il dispositivo, come si scrive in esso un programma o come possono modificare una o più istruzioni: insomma, la gestione da PC.

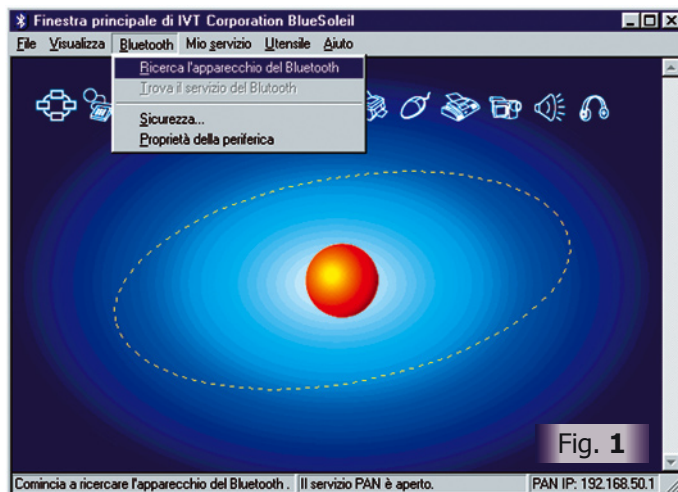
Instaurare la comunicazione

Quando viene messo in funzione per la prima volta, il nostro apparecchio si accende e aspetta di ricevere il programma da eseguire; in esso gira un programma essenziale, implementato in fabbrica, che assegna le funzioni di base ai pulsanti e ai led bicolore di cui Mambo dispone. Tra le operazioni avviate dal software precaricato c'è la possibilità di aprire una sessione di comunicazione via Bluetooth, realizzata attivando l'apposita interfaccia e verificando quando essa comunica la richiesta da parte di un computer presente nel raggio di copertura. Non appena viene rilevata una richiesta di comunicazione, il programma avvia il dialogo. Per comunicare con il dispositivo occorre un computer provvisto di interfaccia Bluetooth standard, ovvero equipaggiato con un "dongle" USB, nel quale sia stato installato un apposito programma di gestione di periferiche Bluetooth;

a riguardo va detto che normalmente i costruttori forniscono, insieme all'hardware e ai driver corrispondenti, anche un semplice software di comunicazione. Nel nostro caso, le prove sono state eseguite usando BluSoleil (versione 1.4.9.5) della IVT Corporation, la cui interfaccia utente è illustrata nella Figura 1.



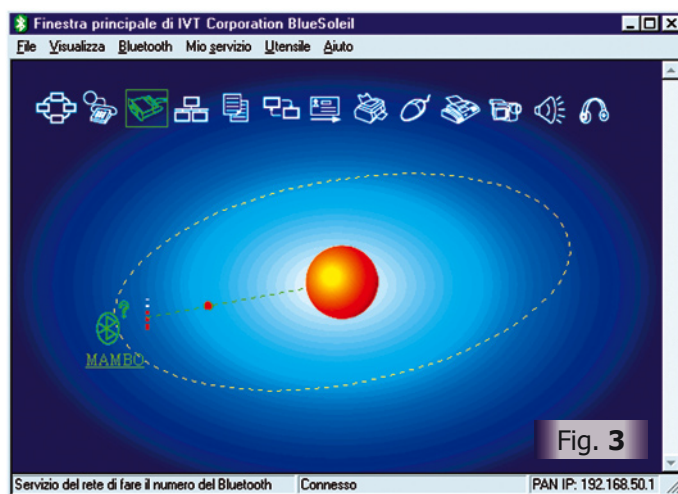
Per cercare i dispositivi attivi nel campo di copertura, bisogna aprire il menu Bluetooth e fare clic sul comando Ricerca dispositivi Bluetooth (Figura 2); se non è stata ancora aperta una COM virtuale, il computer chiede di farlo, altrimenti procede e mostra, al termine della ricerca, le icone corrispondenti agli apparati rilevati. In questa fase,



il programma BlueSoleil esegue una ricerca senza attivare la comunicazione. Per attivarla, bisogna prima

assegnare una COM virtuale al dispositivo, tramite un comando di assegnazione della seriale; quindi si sceglie l'indirizzo di COM (3, 4, 41...) fra quelli liberi. Tale procedura viene effettuata dall'interfaccia utente del programma di dia-

logo Bluetooth. Nel caso di BlueSoleil, l'apertura della COM virtuale si effettua facendo clic, nella finestra di dialogo principale di Figura 1,



sul nome del dispositivo trovato al quale si intende assegnarla; a questo punto risulta evidenziato il pulsante corrispondente al servizio della porta seriale (l'icona con il connettore seriale) sul quale bisogna fare clic per accedere alla finestra di dialogo con cui il computer comunica con la COM aperta e chiede se deve procedere alla connessione. Facendo clic sul pulsante "Sì" viene impartito il comando che avvia la comunicazione: entro alcuni secondi, accanto all'icona dal nome Mambo visibile nella finestra di lavoro del programma (che visualizza i dispositivi trovati nel campo di copertura del Bluetooth), deve apparire un tratteggio rosso verticale e un punto dello stesso colore che si sposta verso la zona centrale dello schermo (Figura 3); ciò significa che la comunicazione è stata instaurata.

Il workbench

Falcom, produttore di Mambo, ha realizzato un software applicativo per ambiente Microsoft Windows (98, 2000, XP) ideale per svolgere tutte le operazioni di programmazione, trasferimento dei dati, acquisizione della memoria riguardante le coordinate del ricevitore GPS, ecc. Il programma in questione si chiama workbench (letteralmente, banco da lavoro) e viene fornito su CD-ROM nella stessa confezione di Mambo. L'interfaccia utente del workbench

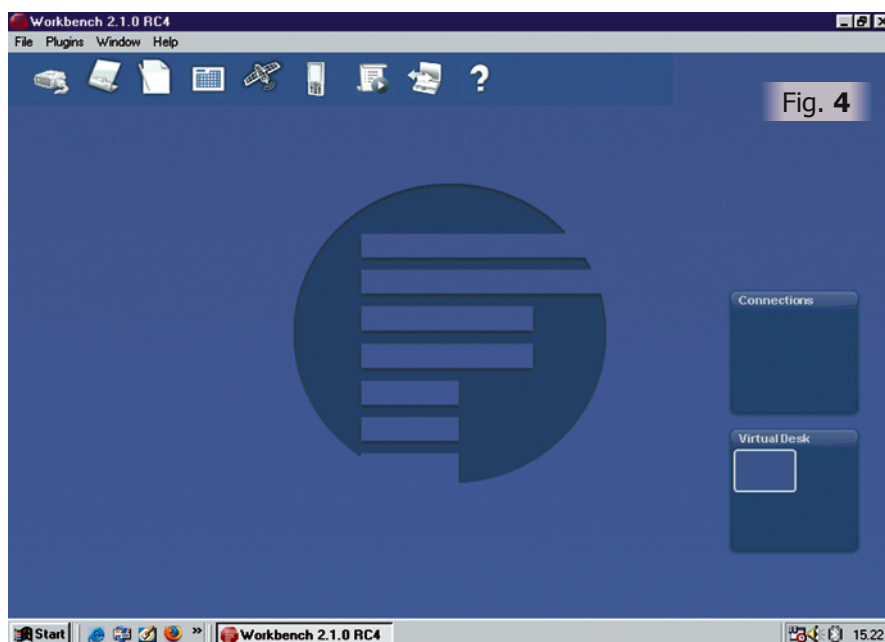


Fig. 4

Falcom (riferita alla versione 2.1.0 RC4) è illustrata nella Figura 4; la finestra di lavoro contiene alcuni menu a tendina e delle icone, corrispondenti ciascuna a una precisa funzione. Le icone sono in realtà i pulsanti della barra degli strumenti, che riassume le principali funzioni; facciamo osservare che i comandi corrispondenti possono essere impartiti anche dal menu *Plugins*.

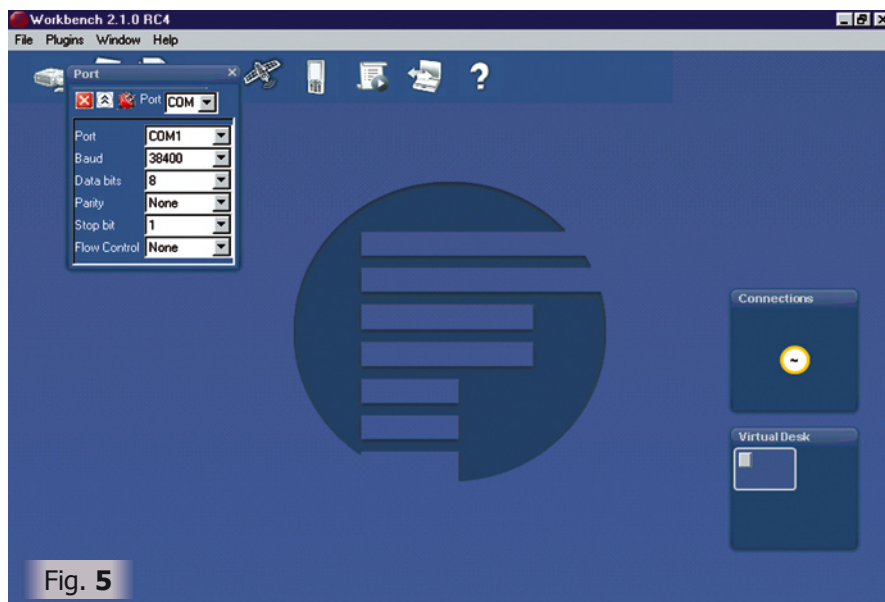
Nella finestra principale del Workbench sono collocate due sezioni chiamate *Virtual Desk* e *Connections*; la prima, situata in basso a destra nella finestra principale, rappresenta un'estensione dell'intero desktop, che si estende ben oltre l'area visualizzata nello schermo del computer.

Virtual Desk è, insomma, una miniatura dell'effettivo piano di lavoro e le finestre aperte (corrispondenti alle funzioni del workbench) vengono visualizzate in essa come icone che seguono la posizione delle finestre vere e

proprie; il rettangolo corrisponde all'area del desktop visualizzata nello schermo. Trascinando con il mouse una finestra da una parte all'altra dello schermo, si vede la sua icona in miniatura che si sposta lungo la finestra del Virtual Desk; aprendo una nuova finestra, ovvero una nuova funzione, la finestra precedente viene spostata fuori dal rettangolo. Insomma, Virtual Desk può essere considerato un monitor di quello che si fa lavorando con il workbench, tuttavia è qualcosa di più: è uno strumento interattivo agendo sul quale, con il puntatore del mouse, si possono spostare le finestre aperte direttamente sullo schermo, senza agire sul desktop reale; sempre in Virtual Desk, si può passare da una finestra all'altra, semplicemente facendo clic sulla miniatura corrispondente. L'altra sezione (*Connections*) è molto più importante perché riassume in un riquadro le connessioni realizzate o realizzabili; in altre parole, riassume le funzioni aperte e i dispositivi Mambo attualmente rintracciati dal programma di comunicazione Bluetooth e in collegamento con il Workbench. È importante perché è in essa che le funzioni si assegnano a uno dei dispositivi trovati, ovvero che le si abbina alla COM virtuale



corrispondente a una sessione di comunicazione aperta con un Mambo. *Connections* serve essenzialmente quando più dispositivi Mambo sono connessi al computer; in tal caso permette di passare agevolmente dall'uno all'altro, ovvero di sconnettere, per esempio, l'editor da un Mambo per assegnarlo ad un altro. Alla luce di quanto detto riguardo alla sezione *Connections*, la si può considerare "la procedura standard di lavoro con Workbench": avviato il programma, per prima cosa bisogna decidere quale delle funzioni attivare, cioè se si desidera operare un editing (con Editor) o inviare comandi direttamente a un Mambo; fatto ciò, si avvia la funzione corrispondente dalla barra degli strumenti. Quando appare sullo schermo la relativa finestra di dialogo, ci si porta nella sezione *Connections*, che deve mostrare un cerchio rosso con il nome della funzione in miniatura. Ora, se esistono COM virtuali aperte e assegnate al programma, sempre nello stesso riquadro le si vede sotto forma di cerchi gialli; in caso contrario, bisogna avviare la procedura di apertura di una porta, definendo in essa il numero della COM usata dal dispositivo Bluetooth per gestire la comunicazione con il Mambo da controllare con Workbench. Il numero della COM in questione si ricava dal programma di gestione



dell'interfaccia Bluetooth (nel nostro caso, BlueSoleil).

Infine, individuata la porta virtuale e atteso che la relativa icona gialla appaia in *Connections*, basta fare clic sul cerchio rosso corrispondente alla funzione da avviare; a questo punto il Workbench conferma l'assegnazione della funzione alla sessione di comunicazione relativa al Mambo desiderato, facendo divenire verde il cerchio. La procedura può sembrare complessa quindi passiamo subito a vedere le funzioni del Workbench, una ad una, iniziando proprio da quella che viene per prima sia nella barra degli strumenti, sia nella sequenza di lavoro con i Mambo.

Selezione porta

Prima di avviare ogni sessione di lavoro in Workbench, bisogna iniziare il dialogo via Bluetooth; pertanto, dopo aver avviato il software Falcom occorre, innanzitutto, accedere alla funzione *Open a new serial port view*, semplicemente facendo clic sulla prima icona di sinistra che appare nella schermata principale. Si accede così alla relativa finestra di dialogo (Figura 5) che riepiloga le porte disponibili e le impostazioni (baud-rate, bit di parità e di dati, ecc.) corrispondenti; in essa l'utente deve scegliere la COM che, dalle

informazioni ricavate dal programma di gestione del Bluetooth, corrisponde alla porta virtuale che il computer ha assegnato all'interfaccia BT del Mambo. La COM virtuale è stata definita nel programma di gestione del BT (nel nostro caso il Blue Soleil) al momento dell'avvio del servizio e della registrazione dei dispositivi.

Definita la porta da usare per interfacciare il Workbench con l'ambiente Bluetooth e impostati i parametri della comunicazione, per stabilire la connessione bisogna fare clic sul terzo (da sinistra) dei pulsanti (*Open Port*) della finestra di dialogo *Port*, quello rosso che somiglia a una spina della corrente. Da questo momento il Workbench si affaccia sulla rete di dispositivi Bluetooth usando l'interfaccia del computer; nella sezione *Connections*, la COM aperta appare come un cerchio giallo. Per chiudere una COM virtuale collegata al Workbench si può procedere in due modi: fare nuovamente clic sul pulsante con la spina o chiudere la finestra di dialogo corrispondente alle impostazioni delle COM. Alla luce di ciò, si badi che una volta aperta la porta, se si intende lavorare con essa non si può chiudere la finestra aperta con *Open a new serial port view*; bisogna lasciarla aperta.



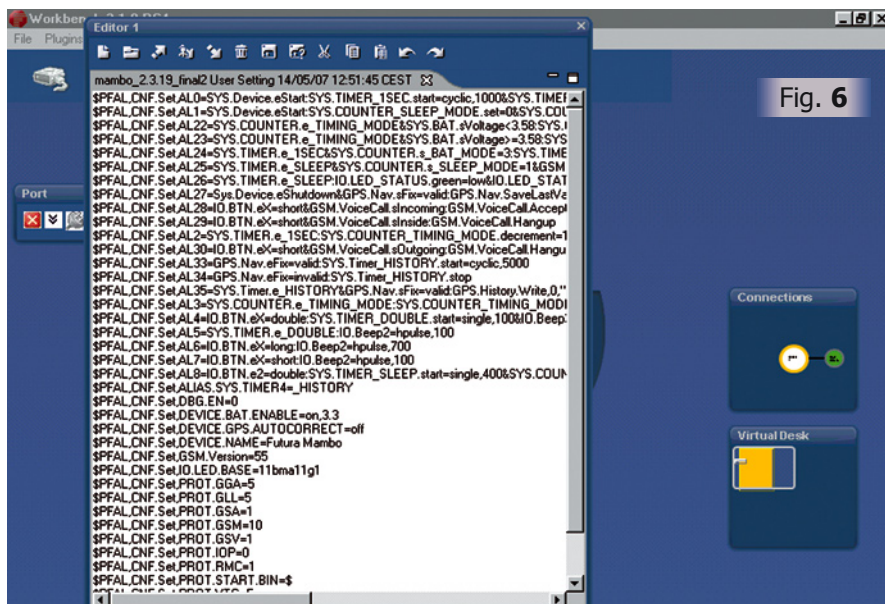


Fig. 6

Editor

Si tratta di una sorta di programma di creazione e modifica di testi, solo che è destinato alla visione, alla scrittura e alla modifica del programma diretto verso il Mambo. Più esattamente, *Editor* permette di vedere il programma acquisito da un dispositivo Mambo, apportare in esso delle modifiche, oppure scrivere ex novo un programma di lavoro.

Per acquisire il programma da un Mambo, oppure inviare a quest'ultimo un nuovo listato o una modifica, si deve passare dalla Console; è tuttavia possibile trasmettere una singola riga di comando semplicemente facendo doppio clic sulla riga corrispondente, quindi evitando di passare dalla Console.

Vediamo i comandi contenuti nella finestra di dialogo dell'*Editor*, analizzando i relativi pulsanti collocati in alto, a partire da sinistra: il primo (*Open New Editor*) si comprende sapendo che *Editor* è una funzione multisessione, nel senso che si possono aprire più sessioni di editing così da poter lavorare su più programmi senza dover necessariamente chiudere un listato. Ciò è di grande utilità quando, ad esempio, si deve scrivere una parte di programma destinata a un Mambo copiandola da quello di un altro dispositivo.

Ebbene, il pulsante in questione permette di aprire una nuova sessione di editing.

Il secondo comando è *Open Configuration File* e serve ad aprire un file di configurazione salvato in una delle periferiche di memorizzazione del computer; appare la finestra di dialogo "Risorse del computer" dalla quale l'utente può indicare qual è il file contenente la configurazione da caricare per editarla ed eventualmente inviarla al Mambo connesso. Il terzo comando è *Start Sending Configuration*: facendo clic sul pulsante corrispondente, il Workbench invia il listato (quindi il programma aperto nell'*Editor*) al Mambo connesso; se l'utente non ha definito alcuna connessione, il programma risponde con una finestra di avviso contenente il messaggio *Editor is not Connected. Please check connections*. È quindi necessario andare nella sezione *Connections* e indicare quale dispositivo va connesso all'*Editor*.

Va notato che impartendo il comando in questione la nuova configurazione inviata prende il posto di quella esistente, nel senso che la memoria di configurazione del Mambo viene cancellata e sostituita con la configurazione scritta nell'editor.

Facendo clic sul pulsante *Read User*

configuration, si impartisce il quarto comando; con esso il software ordina al computer di interrogare il dispositivo Mambo connesso e, ad operazione ultimata, presenta nella finestra dell'*Editor* il listato di configurazione letto dalla memoria di programma del Mambo attualmente connesso (Figura 6).

Il pulsante accanto serve a impartire il comando che dà la configurazione del dispositivo attualmente in uso.

Il sesto dei pulsanti (ha un cestino per icona...) serve a cancellare il contenuto della finestra dell'*Editor*; va usato quando si intende rimuovere le righe di comando scritte per ricominciare da capo. Si noti che la rimozione di un listato, caricato con il comando *Open Configuration File*, dalla finestra dell'*Editor* non comporta l'eliminazione del file dall'hard disk.

Il comando *Save Current Editor Content* (settimo pulsante) apre la solita finestra di dialogo "Risorse del computer", e permette di definire in quale posizione (unità a disco, cartella) salvare il listato di configurazione correntemente aperto nell'*Editor*, oltre al nome che si desidera dare al relativo file.

Il pulsante accanto corrisponde alla funzione Windows "salva con nome" e apre una finestra di dialogo uguale a quella che nel sistema operativo corrisponde a tale funzione. ➤



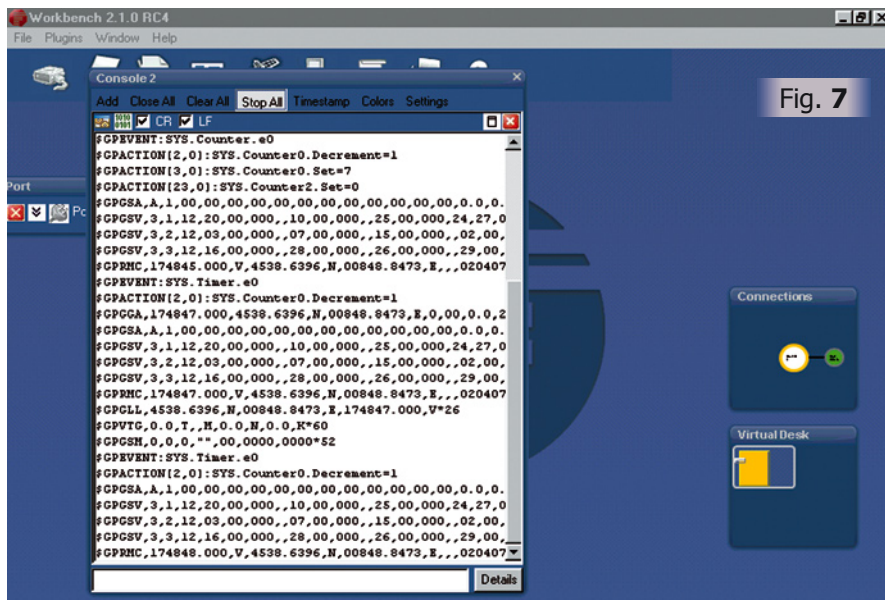


Fig. 7

I pulsanti che restano servono a svolgere le funzioni tipiche di un editor di testo, ossia *Taglia*, *Copia*, *Incolla*, *Annulla modifica* e *Annulla l'operazione annullata*.

Console

Spiegato l'*Editor*, passiamo adesso alla funzione *Console*: si attiva facendo clic sul pulsante omonimo del desktop del Workbench e serve per dialogare con i dispositivi Mambo; sostanzialmente, è assimilabile alla finestra di un programma di Terminale, quindi da essa si possono impartire comandi che raggiungeranno direttamente il dispositivo Mambo attualmente connesso. Per comandi si intende non quelli inerenti alla configurazione del funzionamento (le istruzioni), ma quelli riguardanti la comunicazione, l'invio e la lettura di file di configurazione contenuti nella memoria dei Mambo, l'acquisizione della memoria riguardante i punti e le coordinate GPS ed altro ancora. Perché avvenga il dialogo, bisogna prima definire, nella solita sezione *Connections*, a quale Mambo si vuole assegnare la sessione di lavoro corrispondente alla *Console* aperta. Una volta assegnata la funzione, nella finestra scorrono tutti gli eventi trasmessi dalla memoria del Mambo attualmente connesso (Figura 7); è

possibile arrestare lo scorrimento con il pulsante *Stop All*.

History

Il quarto pulsante della barra degli strumenti corrisponde alla funzione *History*, grazie alla quale è possibile operare sulla memoria degli eventi (storico) del dispositivo Mambo attualmente connesso (Figura 8); come per le altre funzioni, l'assegnazione del Mambo con cui si intende lavorare si effettua dalla sezione *Connections*, facendo clic sull'icona corrispondente al dispositivo di interesse. Accedendo a *History* è possibile scaricare l'elenco cronologico completo

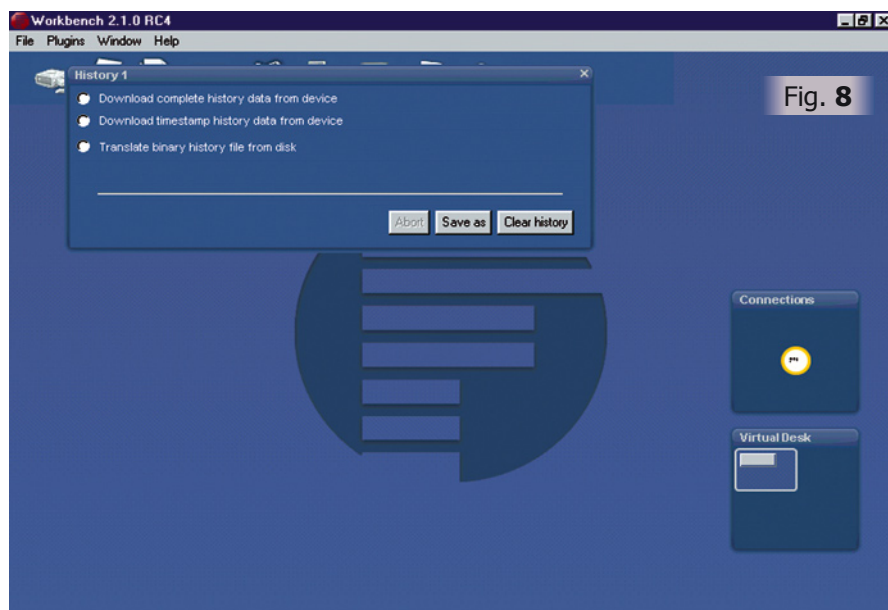


Fig. 8

degli eventi registrati dal Mambo, oppure un elenco, ancora cronologico, dei soli eventi riguardanti le operazioni di comunicazione, le segnalazioni di stato (livello del segnale, satelliti e via di seguito).

La differenza tra le due operazioni sta nel fatto che una permette di acquisire tutte le situazioni riguardanti l'attività del Mambo attualmente connesso, quindi tutti i dati riguardanti il funzionamento, che possono includere il funzionamento della connessione telefonica GSM/GPRS (gestore, livello del segnale, comunicazioni ecc.) quello del ricevitore satellitare GPS (stato dei satelliti agganciati, livello del segnale, coordinate scaricate, stringhe NMEA) stato della connessione Bluetooth, mentre l'altra estrae solo parte dei dati.

In altre parole, si può definire l'intervallo temporale (da un certo giorno a un altro) da prendere in considerazione, ovvero scaricare tutto quanto è stato registrato, fermi restando i limiti determinati dalla capienza della memoria eventi del Mambo; a tale riguardo si noti che quando la memoria è piena il dispositivo sovrascrive i record iniziando da quelli più vecchi, nel senso che cancella le posizioni più datate per scrivere in esse i nuovi eventi.

Una terza opzione ci consente di

trasferire uno storico degli eventi memorizzato in un'unità a disco del computer nella memoria del Mambo attualmente in connessione, ovvero di salvare un log degli eventi su disco (pulsante *Save as*). Ancora, tramite un apposito pulsante (*Clear History*) si può impartire al Mambo l'ordine di cancellare completamente lo storico degli eventi.

Allo scopo di agevolare la consultazione dell'archivio storico degli eventi, il Workbench prevede la possibilità di applicare dei filtri di vario genere, che consentono di selezionare il tipo di dati riepilogati, ma anche di andare a cercare determinate stringhe o singoli dati specifici.

Una particolarità interessante della funzione *History* è la possibilità di scaricare dalla memoria del Mambo selezionato i dati inerenti alla localizzazione, ovvero le coordinate rilevate, in ordine cronologico; in questo caso l'utente può decidere se salvare nel computer i record corrispondenti in formato binario (formato macchina) o sotto forma di stringhe NMEA0183 intere: nel primo caso il Workbench memorizza tutti i dati contenuti in ciascuna stringa inviata dal Mambo, mentre nel secondo estrae i soli dati riguardanti le coordinate. L'utilizzo del file in formato NMEA0183 è utile quan-

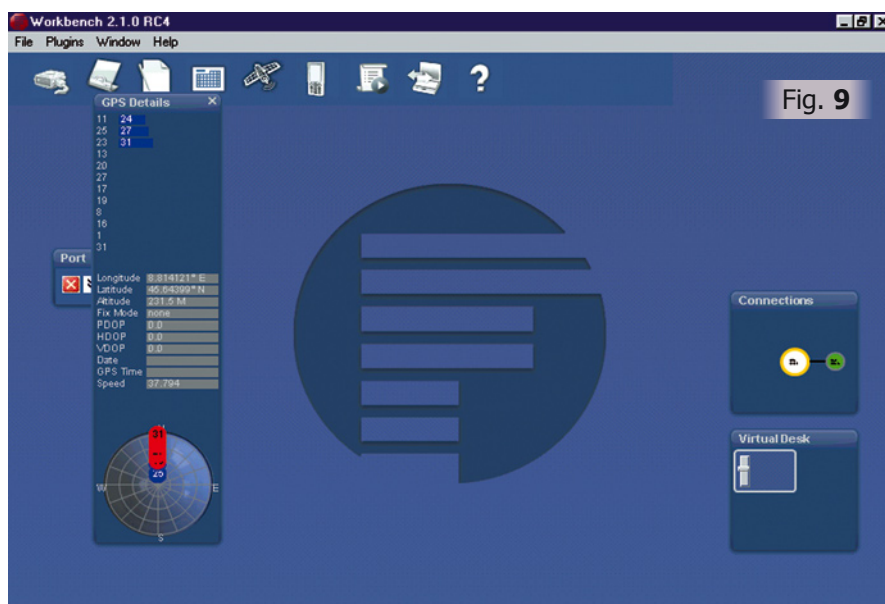


Fig. 9

do si intende inviare subito i dati a un software di navigazione e vedere sulla cartografia del caso il percorso compiuto dal localizzatore. Va comunque detto che ciò è possibile anche scegliendo di salvare il file intero della memoria, in formato binario; in tal caso si sfruttano le funzionalità di alcuni software di navigazione satellitare che permettono di estrarre le coordinate dai dati in formato macchina, ovvero di ricavare le informazioni secondo il protocollo NMEA0183 dal file binario.

Informazioni GPS

La quinta icona della barra degli

strumenti dà accesso alla funzione *GPS Detail* e all'omonima finestra di notifica (Figura 9) in cui il programma riepiloga la situazione dei satelliti dei quali il Mambo riceve il segnale; in essa si trovano informazioni sulla posizione, il livello del segnale, la modalità di funzionamento, lo stato del Fix, l'orario, l'eventuale velocità del veicolo sul quale si trova il Mambo (ammesso che sia in movimento).

Come per tutte le altre funzioni, anche questa si assegna a un determinato dispositivo Mambo selezionandolo nella sezione *Connections*, nel modo consueto, già spiegato in precedenza.

Informazioni sul cellulare

Un comando (*GSM Detail*) analogo al precedente, ma riservato alla sezione GSM/GPRS del Mambo, si impartisce facendo clic sulla sesta icona della barra degli strumenti; permette di accedere a una finestra (Figura 10) che riepiloga le condizioni dell'eventuale connessione telefonica radiomobile del dispositivo selezionato mediante la sezione *Connections*.

La finestra cui si accede contiene le informazioni sul livello del segnale, il nome del gestore (o il suo codice identificativo) della rete cui il di-

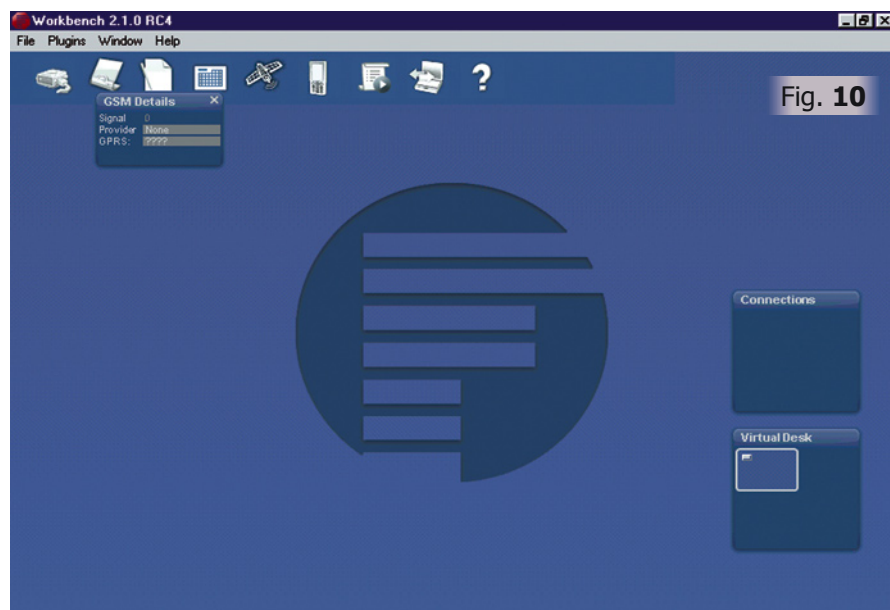


Fig. 10

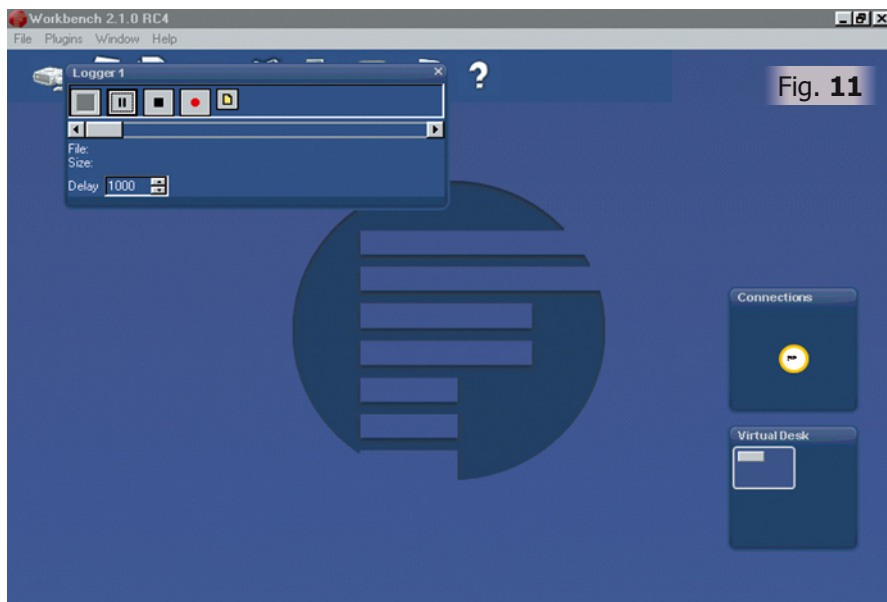


Fig. 11

positivo è agganciato e l'eventuale presenza dell'attività GPRS.

Il Logger

La settima icona della barra degli strumenti corrisponde al comando *Logger* e avvia una sorta di registratore di eventi, i cui parametri di funzionamento possono essere definiti nell'apposita finestra di dialogo cui si accede (Figura 11); il registratore crea un file che memorizza tutto quel che accade durante la connessione con il Mambo abbinato mediante la solita procedura che passa dalla sezione *Connections*.

Informazioni I/O

L'ultima funzione del Workbench che analizziamo è quella chiamata *I/O Detail* alla quale si accede facendo clic sull'ottava icona della barra degli strumenti, ovvero impartendo l'omonimo comando del menu *Plugins*. Dà accesso a una finestra di notifica (Figura 12) che riepiloga le condizioni dei comandi locali e della batteria del Mambo (attualmente connesso mediante la solita sezione *Connections*).

Con questo termina la descrizione delle funzioni del programma Falcom; chi volesse maggiori dettagli, potrà trovarli nell'*Help* in li-

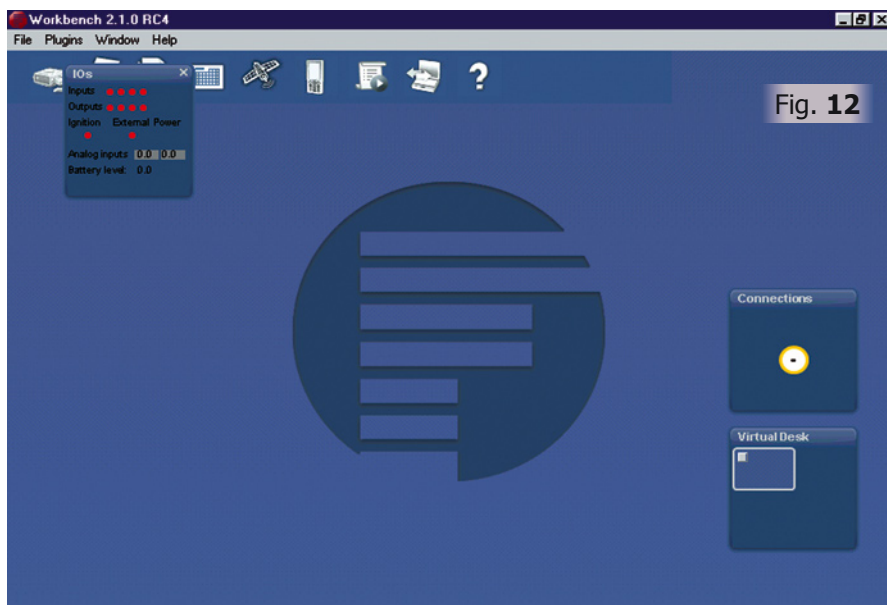


Fig. 12

nea accessibile facendo clic sul pulsante più a destra della barra degli strumenti (quello contraddistinto dal punto interrogativo) che costituisce un valido aiuto all'operatore che, durante una sessione di lavoro, avesse qualche dubbio sull'utilizzo di una certa funzione o sulla definizione di un parametro.

Il file di configurazione

Bene, esaminato il programma Workbench e le sue principali funzioni, possiamo spostarci fisicamente in Mambo e andare a vedere come è strutturato il file di configurazione, cioè quell'insieme di comandi che dice al dispositivo come deve comportarsi.

Partiamo dicendo che inizialmente, ossia all'uscita dalla fabbrica, nella memoria viene scritto un file come quello visibile nel Listato 1; esso contiene le istruzioni di base, che servono a gestire i pulsanti al fine di accendere e spegnere Mambo, instaurare la comunicazione con il computer, gestire le chiamate telefoniche e dare le segnalazioni del caso. Proviamo ad esaminare alcuni punti del listato e vediamo che all'accensione il software inizializza due timer ciclici, uno da 1 secondo e l'altro da 20, oltre ad un contatore da 7 secondi; configura il GPS perché sia acceso con un intervallo on/off di 400/1.000 ms.

I timer servono per valutare e segnalare l'attività del GPS, oltre che per il controllo di altre funzioni quali la connessione telefonica, lo stato di carica della batteria e l'accensione dei led che danno le segnalazioni visive del caso.

I pulsanti servono, oltre che per accendere e spegnere Mambo, alla gestione delle chiamate telefoniche, che il dispositivo, per impostazione predefinita, può sia effettuare che ricevere; in particolare, la pressione breve di uno qualsiasi di essi determina la sospensione di una telefonata in corso, sia essa ricevuta o

Listato 1

```
$PFAL,CNF.Set,AL0=SYS.Device.eStart:SYS.TIMER_1SEC.start=cyclic,1000&SYS.TIMER_20SEC.start=cyclic,20000&SYS.COUNTER_TIMING_MODE.set=7&GPS.Nav.PowerSave=on,400,1000
All'accensione inizializza un timer (TIMER_1SEC) ciclico della durata di un secondo, un timer dello stesso tipo della durata di 20 sec (TIMER_20SEC), un contatore a 7 e configura il GPS in modo che stia acceso 400 ms su 1000 ms.

$PFAL,CNF.Set,AL1=SYS.Device.eStart:SYS.COUNTER_SLEEP_MODE.set=0&SYS.COUNTER_BAT_MODE.set=0
Inizializza il contatore COUNTER_SLEEP_MODE a 0 e il contatore COUNTER_BAT_MODE a 0

$PFAL,CNF.Set,AL2=SYS.TIMER.e_1SEC:SYS.COUNTER_TIMING_MODE.decrement=1
Se il TIMER_1SEC va a 0 (dopo quindi un secondo) allora (:) decrementa il contatore COUNTER_TIMING_MODE di 1

$PFAL,CNF.Set,AL3=SYS.COUNTER.e_TIMING_MODE:SYS.COUNTER_TIMING_MODE.set=7
Se il contatore COUNTER_TIMING_MODE arriva a 0 resettalo a 7

$PFAL,CNF.Set,AL4=IO.BTN.eX=double:SYS.TIMER_DOUBLE.start=single,100&IO.Beep3=hpulse,200
Se uno qualsiasi dei pulsanti è premuto due volte (entro 500 ms) allora inizializza il TIMER_DOUBLE ad un valore di 100 ms e emetti un beep

$PFAL,CNF.Set,AL5=SYS.TIMER.e_DOUBLE:IO.Beep2=hpulse,100
Se il TIMER_DOUBLE va a 0 (dopo quindi 100ms) allora emetti un beep

... descrizione più approfondita nella prossima puntata.
```

effettuata dall'apparato. Inoltre, se è in arrivo una chiamata, premendo brevemente uno dei tasti si risponde ad essa. Riguardo allo spegnimento, va detto che Mambo, se il comando viene dato quando la sezione GPS è in fix, prima di spegnersi va a salvare nella propria memoria l'ultimo punto valido. Del file di configurazione va detto che è composto da due porzioni, una modificabile dall'utente accedendo dalle apposite funzioni del Workbench e l'altra contenente parametri che non possono essere modificati: la prima è detta *User Configuration* e la seconda *Device Configuration*. Quindi si può andare a sovrascrivere la sola *User Configuration*. Restando in tema di programmazione di Mambo, in questa puntata vogliamo farvi vedere un altro esempio di configurazione mirata a determinate funzioni; per esempio, il Listato 2 è uno *User Configuration* che comprende i comandi da inviare per definire la modalità di funzionamento dei led bicolore e la sensibilità della sezione GPS. Tenete presente che nell'esempio in questione, avente scopo prettamente didattico, sono stati rimossi alcuni parametri predefiniti nello *User Configuration* inserito in fabbrica nella memoria di Mambo.

Nello specifico, con i comandi pre-

senti nel listato si stabiliscono le temporizzazioni inerenti all'invio delle stringhe GGA, GLL, GSA, GSM, GSV, RMC e GTV e la modalità automatica per i led; si definisce altresì il nome con cui il dispositivo deve identificarsi, cosa necessaria quando ci si trova a dover ricevere da una singola postazione i dati di più Mambo collocati su diversi veicoli o indossati da più persone. Per

quel che riguarda la sezione GPS, vengono disattivate la correzione e il modo Power-Save (quindi il timer 400/1.000 ms. che per impostazione predefinita ne controlla attivazione e disattivazione) perciò il ricevitore è "sempre acceso".

Nella prossima puntata analizzeremo in dettaglio i file di configurazione, quindi non perdetevi il prossimo fascicolo.

Listato 2

```
$PFAL,CNF.Set,PROT.GGA=5
Imposta l'invio della stringa GGA ogni 5 secondi
$PFAL,CNF.Set,PROT.GLL=5
Imposta l'invio della stringa GLL ogni 5 secondi
$PFAL,CNF.Set,PROT.GSA=1
Imposta l'invio della stringa GSA ogni secondo
$PFAL,CNF.Set,PROT.GSM=10
Imposta l'invio della stringa GSM ogni 10 secondi
$PFAL,CNF.Set,PROT.GSV=1
Imposta l'invio della stringa GSV ogni secondo
$PFAL,CNF.Set,PROT.RMC=1
Imposta l'invio della stringa RMC ogni secondo
$PFAL,CNF.Set,PROT.VTG=5
Imposta l'invio della stringa VTG ogni 5 secondi
$PFAL,Cnf.Set,IO.LED.BASE=11bma11g1
Imposta i led per funzionare in modo automatico seguendo l'impostazione delle periferiche
$PFAL,CNF.Set,DEVICE.NAME=Elettronica In Mambo
Imposta il nome del Mambo
$PFAL,CNF.Set,DEVICE.GPS.AUTOCORRECT=off
Disabilita la correzione del GPS
$PFAL,CNF.Set,DEVICE.BAT.ENABLE=on,3,3
Abilita il monitoraggio della tensione di ricarica e se supera i 3,7V viene richiamato l'evento Sys.Power.ePlugged
$PFAL,CNF.Set,AL0=SYS.Device.eStart:SYS.TIMER_1SEC.start=cyclic,1000&SYS.TIMER_20SEC.start=cyclic,20000&SYS.COUNTER_TIMING_MODE.set=7&GPS.Nav.PowerSave=off,900,1000
Disabilita il PowerSave del GPS
$PFAL,CNF.Clear,"AL9"
...
$PFAL,CNF.Clear,"AL21"
Cancella le impostazioni relative all'accensione dei led per l'indicazione dello stato della batteria
```


Radiocomando 433,92 MHz a 4 canali

di *Francesco Doni*

Nuova decodifica quadricanale che consente di realizzare semplicemente un sistema di controllo remoto a 433 MHz a quattro canali con uscita a relè. Dispone di 1.024 combinazioni e garantisce una portata di circa 50 metri in assenza di ostacoli. I relè di uscita possono operare sia in modalità bistabile che in modalità astabile e sono adatti per carichi fino a 1A.

A partire dall'ultimo decennio del secolo scorso, l'evoluzione dei radiocomandi (e non solo di quelli) ha subito una forte spinta sotto l'effetto di esigenze che sono state e sono essenzialmente la sicurezza e la possibilità di riuscire ad attivare ciò che si desidera, al riparo dall'interferenza di altri sistemi operanti nella stessa zona. Nel primo caso la tecnologia ha risposto con lo sviluppo di encoder sempre più complessi: è stato così possibile ridurre il rischio di interferenze nei sistemi di sicurezza, semplicemente riducendo le probabilità che un radiocomando estraneo al sistema, ma analogo

a quello del legittimo proprietario, potesse trovare il codice d'accesso anche casualmente. Iniziando col realizzare codificatori a numero di combinazioni sempre più elevato, si è poi passati a quelli a codifica variabile (rolling-code) che sono i più attuali e sicuri. Sul piano dell'esclusività del comando, sono stati realizzati radiocomandi a più frequenze entro la stessa banda, così da permettere un'attivazione certa anche in un ambiente in cui si trovano ormai molti apparati radiocomandati ed altrettante persone che li attivano a tutte le ore del giorno. Va comunque detto che non sempre serve ave- ➤

La decodifica D4ML

Si tratta di un ibrido in SMD realizzato dall'Aurel su supporto di allumina, protetto con resina sintetica e provvisto di 20 piedini Single-In-Line a passo 2,54 mm. Contiene un decodificatore integrato (HT12D) capace di riconoscere le stringhe nel formato caratteristico dei classici encoder MM53200/UM3750 o UM86409 (a 4.096 combinazioni) e di attivare, a seconda dei dati in ingresso, due uscite che comandano un decoder binario con quattro uscite. Queste pilotano ciascuna un transistor NPN montato in configurazione open-collector, che funziona da amplificatore di corrente (interruttore statico...) e può alimentare in modo "sink" carichi funzionanti con tensioni fino a 15 Vcc; ogni transistor può assorbire fino a 50 mA. Ogni volta che viene ricevuta una stringa di dati, il decodificatore ne confronta i dieci bit di codifica con gli stati logici impostati sui propri pin 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 (il piedino 1 è l'LSB, mentre il 10 è l'MSB) e, se i dieci dip di indirizzo locale sono impostati esattamente come i corrispondenti del trasmettitore, il decodificatore va a leggere i bit relativi alla codifica del pulsante per scoprire a quale canale si riferisce la trasmissione.

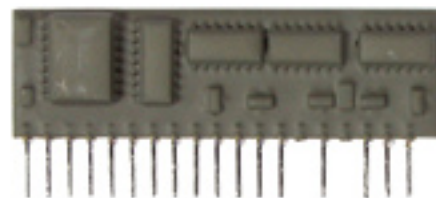
La tabella della verità è conforme a quella dei trasmettitori Aurel TX3750-4C-SAW e, prendendo in considerazione i bit 10 e 11, prevede che il canale 1 corrisponda alla condizione 01 (bit 10 a zero logico e 11 ad uno) e il 2, il

3 ed il 4 alle condizioni, rispettivamente, 10, 11 e 00. Riconosciuto il canale, il decodificatore interno al D4ML comanda il decoder 1 a 4, compito del quale è presentare in uscita il valore corrispondente, attivando una sola uscita alla volta.

Le caratteristiche tecniche del modulo sono le seguenti:

- Combinazioni: 1.024* (10 bit)
- Frequenza di clock dell'encoder: 1 kHz
- Uscite: 4 a transistor open-collector
- Corrente commutabile dalle uscite: 50 mA
- Tensione di lavoro delle uscite: 0÷15 V
- Tensione di alimentazione: 5÷15 Vcc
- Corrente assorbita (a riposo): 50 µA

* Pur essendo possibili 4.096 combinazioni, corrispondenti a 12 bit binari, il sistema ha una codifica a sole 1.024 combinazioni perché due bit vengono usati per contraddistinguere i canali, quindi ne restano solo 10.



re il radiocomando più sicuro, e che non bisogna considerare superato un sistema a codifica tradizionale con "ridotto" numero di combinazioni. Per questo motivo riteniamo ancora attuale utilizzare trasmettitori e ricevitori basati sulla codifica di tipo UM3750. Ecco perché in queste pagine proponiamo il progetto di un ricevitore per radiocomando (il trasmettitore è disponibile in commercio già montato e collaudato) operante nella gamma di frequenza standard di 433,92 MHz, a codifica fissa basata su HT12E e HT12D, integrati che sono sia encoder che

decoder, nati dall'ormai vetusto MM53200, che è stato il capostipite della serie. Tali componenti prevedono una codifica binaria a 12 bit, quindi un massimo di 4096 combinazioni, e possono realizzare telecomandi (a filo o wireless via radio e a infrarossi) a più di un canale: allo scopo, basta codificare ciascuno trasmettendo un codice diverso, tipicamente differenziato negli ultimi due bit. Utilizzando uno o più bit per contraddistinguere i canali nei sistemi pluricanale, le possibili combinazioni della codifica si riducono di conseguenza; così, se un trasmet-

titore monocanale ha 4.096 combinazioni, un TX a due canali (solitamente si usa il bit 12 per distinguere il primo dal secondo canale...) ne ha 2.048 ed un TX a quattro (due bit sottratti e usati per identificare i canali) appena 1.024.

Ma, come accennato, esistono applicazioni nelle quali la sicurezza non è vitale e l'esclusività del comando non viene messa a repentaglio dalla presenza di troppi sistemi simili. Si parla, ad esempio, del radiocomando di utilizzatori in zone aperte e a bassa densità di popolazione, o in locali molto grandi nei quali difficilmente può giungere il segnale di un trasmettitore uguale a quello del radiocomando usato; dell'attivazione a distanza di pompe a bassa tensione per l'agricoltura in campagna, della luce del giardino o della tapparella o tenda da sole motorizzata, di macchinari in capannoni industriali, di pannelli luminosi e tabelloni segnapunti. Ecco, il radiocomando a quattro canali descritto in questo articolo nasce per queste ed altre

- Frequenza di lavoro: 433,92 MHz;
- Portata (in assenza di ostacoli): 50 metri circa;
- Sensibilità del ricevitore: -97 dBm;
- Tensione d'alimentazione ricevitore: 12 ÷ 15 Vcc;
- Corrente assorbita dal ricevitore: 100 mA max;
- Numero di canali: 4;
- Corrente commutabile dai relé: 1 A;
- Tensione commutabile dai relé: 120 Vac o Vdc.
- Codifica: MM53200 ed equivalenti, a 1024 combinazioni

applicazioni simili. Vediamolo dunque nei dettagli, scomponendolo nelle due unità che lo formano: la trasmittente e la ricevente.

L'unità TX

Il trasmettitore è un prodotto abbastanza convenzionale e può essere acquistato già montato e collaudato presso molti negozi di elettronica e siti Internet; opera sui 433,92 MHz irradiando il proprio segnale con una potenza di 1 mW. Per l'abbinamento con la ricevente a quattro canali qui descritta, abbiamo adottato il TX12E-4C-SAW 433 prodotto dall'Aurel; tuttavia nulla vieta di usare trasmettitori di altro tipo, ma solo a patto di rispettare alcune condizioni: la frequenza di lavoro deve essere 433,92 MHz, la codifica uguale a quella prevista e

la frequenza di clock dell'encoder pari a 1 kHz.

Il TX Aurel contiene un encoder HT12E, versione moderna dell'UM3750 (i due sono equivalenti) il cui segnale digitale modula in ampiezza un oscillatore a risonatore SAW, molto stabile e capace di mantenere la frequenza di emissione costante quanto basta per accordarsi perfettamente con il nostro ricevitore (così da garantire la massima portata possibile) ed evitare eccessive deviazioni che nei radiocomandi moderni, quelli omologabili secondo le norme CE, non sarebbero ammesse. Proprio l'ottima stabilità dell'oscillatore e quindi della trasmissione, rende il TX rispondente alle norme europee EN 300220, EN301489 ed EN60950, in special modo per quanto riguar-

da la stabilità ed il contenimento delle emissioni spurie (ossia segnali collaterali a frequenze diverse da quella di centro banda). Il circuito stampato include l'antenna trasmittente, realizzata con una pista calibrata sia in lunghezza che impedenza. Viste le dimensioni contenute, il circuito è stato inserito, insieme alla pila miniatura a 12 V, in un contenitore plastico molto piccolo (43,5x75 mm) usabile anche come portachiavi. Anche il consumo è ridotto: appena 7 milliampere (tipici) in trasmissione, che permettono un'autonomia notevole, la quale può raggiungere un anno di utilizzo normale (qualche attivazione, della durata di un paio di secondi, ogni giorno). Dalle specifiche fornite dal costruttore si nota che, sebbene per ottenere quattro cana- ➤

Una coppia affiatata

Nel sistema descritto in questo articolo, l'encoder del trasmettitore (HT12E) è accoppiato al chip decoder HT12D del ricevitore. Tale chip, prodotto dalla Holtek, è già integrato nel modulo ibrido D4ML, ma, all'occorrenza, può essere anche acquistato da solo (in formato DIL). In ogni caso, sia che venga usato come chip, sia come modulo D4ML, esso necessita di un trasmettitore codificato e di un ricevitore RF. A tal proposito, il telecomando TX12E-4C-SAW 433 e il ricevitore BCNBK sono i compagni ideali dell'HT12D e del D4ML.

TX12E-4C-SAW 433

È un trasmettitore RF a quattro canali di piccole dimensioni. La sezione RF è dotata di un risonatore SAW, il quale garantisce la trasmissione di un segnale perfettamente stabile in tutte le condizioni di utilizzo. È il prodotto giusto per tutte le applicazioni in cui è necessario comandare sistemi posti nelle immediate vicinanze del trasmettitore. Dispone di 2^{10} combinazioni.



Caratteristiche Tecniche

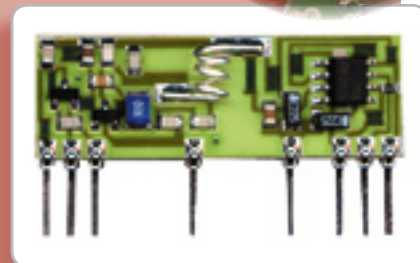
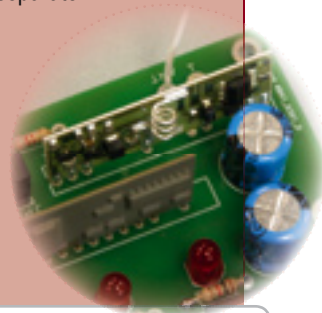
- Encoder: HT12E (UM3750 compatibile).
- Frequenza di clock encoder: 1 kHz.
- Programmazione codice: dip-switch a 10 poli On-Off più dip-switch a 4 poli On-Off.
- Frequenza: 433,92 MHz ottenuta con risonatore SAW.
- Tipo di modulazione: OOK (On-Off Keying).
- Potenza irradiata (E.R.P.): 1 mW (0 dBm).
- Alimentazione: con pila 12 V formato miniatura (V23A).
- Assorbimento tipico: 7 mA (15 mA max.).
- Diodo led per segnalazione di trasmissione in corso.
- Contenitore in plastica antiurto.
- Dimensioni : 75 x 43,5 x 18,5 mm.

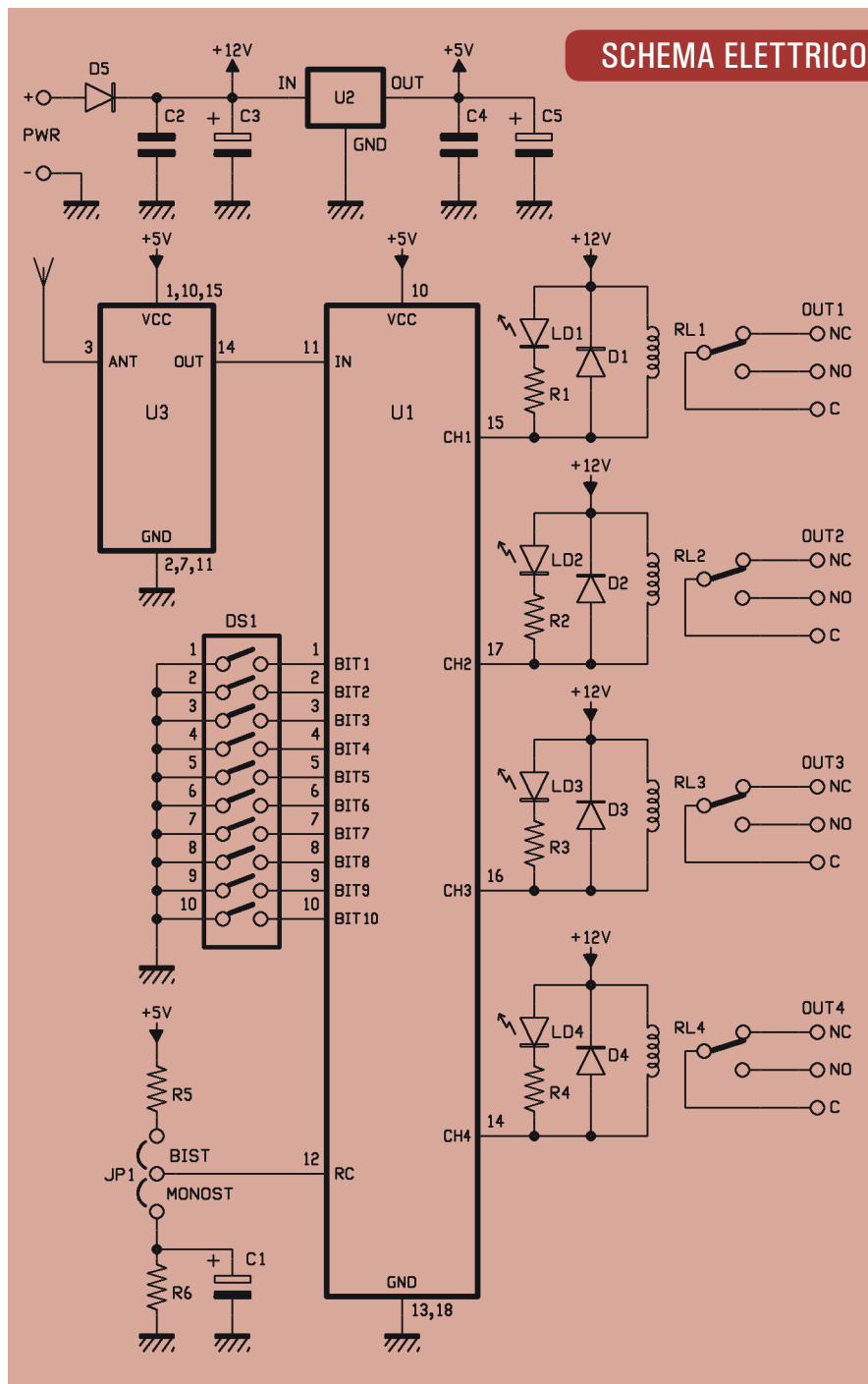
BCNBK

Questo ricevitore offre un eccellente rapporto qualità-prezzo e rispetta le normative vigenti per ciò che concerne sia le emissioni fuori banda che l'impiego di materiali non inquinanti (direttive RoHS). È inoltre pin-to-pin compatibile con il vecchio RF290, dal quale differisce per l'alimentazione singola (l'RF290 dispone di alimentazioni separate per le sezioni RF e BF).

Caratteristiche Tecniche

- Realizzazione: circuito ibrido in allumina
- Dimensioni: 38,1 x 13,7 x 5,5 mm; pin a passo 2,54 mm.
- Frequenza di lavoro: 433,92 MHz.
- Ricezione di segnali modulati OOK (On-Off Keying).
- Sensibilità RF: -97 dBm a centro banda.
- Banda passante RF a -3dB: $\geq \pm 1,2$ MHz.
- Uscita: onda quadra (max. di 2 kHz).
- Uscita normalmente bassa in assenza di segnale RF.
- Alimentazione: +5 V
- Assorbimento: 3 mA (max).
- Emissioni RF spurie: -65 dBm.





costituita anche solo da uno spezzone di filo in rame lungo 17 cm) viene portato al piedino 3 dell'U3, un ricevitore ibrido a 15 pin single-in-line (SIL) a passo 2,54 mm, di tipo BCNBK, che contiene uno stadio ricevente superrigenerativo accordato a 433,92 MHz, al quale segue un demodulatore AM dalla cui uscita si possono prelevare impulsi simili a quelli partiti dall'encoder del trasmettitore; uno squadratore interno (comparatore a trigger di Schmitt) provvede poi a rendere ripidi i fronti del segnale, che esce dal piedino 14. Il componente funziona con un'alimentazione di 5 volt applicata ai piedini 1, 10 e 15. I pin 2, 7 ed 11, invece, sono da collegare a massa. Come tutti i circuiti superrigenerativi, il nostro ricevitore è molto sensibile (-100 dBm, corrispondenti ad appena 2,24 microvolt!) quindi riceve segnali anche abbastanza deboli, quale quello irradiato nell'etere dal trasmettitore portatile, garantendo una buona copertura; presenta però il difetto dei ricevitori in superreazione: è poco selettivo, quindi tenderebbe a ricevere anche trasmissioni su frequenze relativamente molto vicine a quella di accordo. Per restringere la banda ed evitare che interferenze possano disturbare il funzionamento del radiocomando, tra l'ingresso d'antenna (piedino 3) e l'amplificatore RF che precede lo stadio superrigenerativo di sintonia, l'BCNBK dispone di un filtro ceramico che limita la banda passante e quindi la selettività a 2 MHz (± 1 MHz). L'uscita dello stadio a radiofrequenza è collegata al demodulatore d'ampiezza che estrae una componente di bassa frequenza corrispondente a quella generata dal codificatore del trasmettitore, componente che, essendo affetta da un residuo di RF e comunque non molto regolare, viene normalmente fatta passare dallo squadratore del segnale di uscita, che al piedino 14 dà impulsi netti, con fronti di salita

li usualmente si giochi sugli ultimi due bit, nel caso dei trasmettitori usati in abbinamento con il nostro RX la codifica dei canali si compie variando lo stato dei bit 10 e 11.

Il ricevitore

L'unità ricevente è un po' più grande e complessa della trasmittente, come mostra l'apposito schema elettrico visibile nella pagina qui accanto: consta di un radoricevitore e

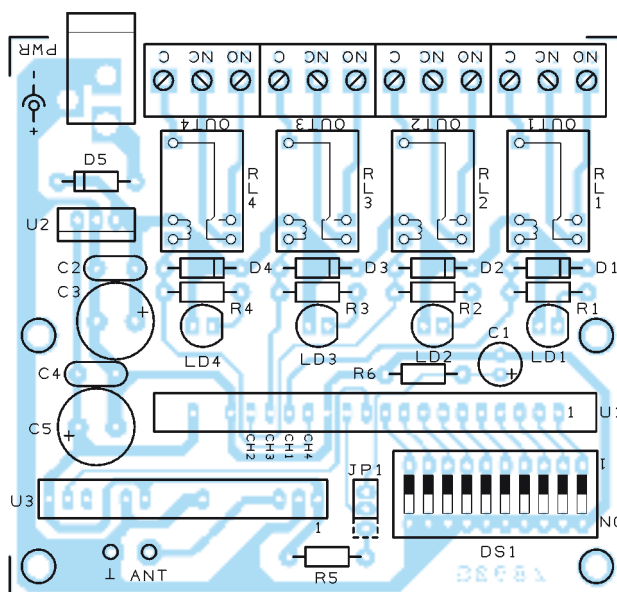
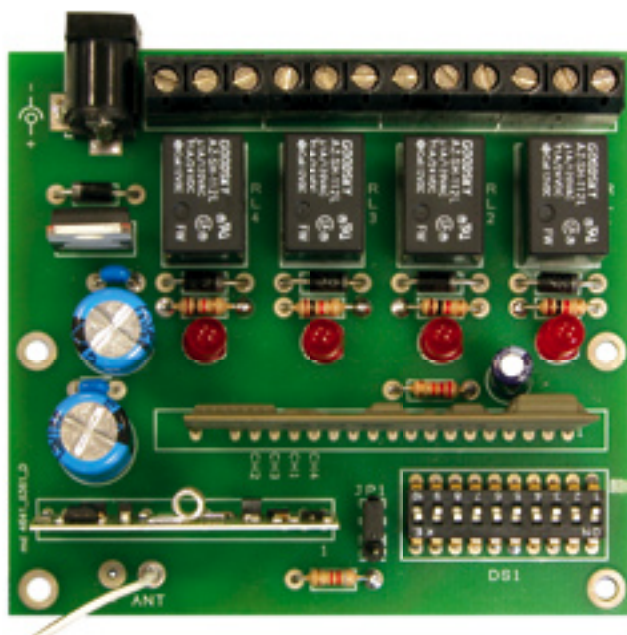
una decodifica, entrambi in tecnologia ibrida, oltre che di un regolatore di tensione integrato, quattro relé, un dip-switch a 10 poli e pochi altri elementi attivi e passivi di contorno. I due ibridi sono prodotti dall'Aurel, casa italiana che si occupa di moduli SMD e ibridi destinati all'automazione e alla comunicazione via radio da oltre vent'anni. Esaminiamo dunque lo schema elettrico: il segnale radio captato dall'antenna (che può essere

molto ripidi e periodi di ampiezza costante. Il segnale contenente il codice digitale ben “ripulito” viene inviato direttamente all’ingresso del decodificatore U1, un circuito ibrido in formato SIL a 20 piedini (passo 2,54 mm) chiamato D4ML, contenente un decodificatore in grado di riconoscere la codifica del caso, un decoder binario “1 a 4”, una logica di controllo e temporizzazione delle uscite e quattro transistor NPN montati in configurazione open-collector che funzionano da amplificatori di corrente e permettono di comandare direttamente utilizzatori alimentati con tensioni fino a 15 Vcc e che assorbano correnti contenute in 50 mA.

Come funziona il decoder

Il decodificatore principale è un integrato dedicato in grado di riconoscere e decifrare i codici a base HT12E; riceve il segnale digitale al piedino 11, e interpreta come impulso il livello logico alto uscente dall’encoder posto sul trasmettitore (nel nostro caso non ci sono problemi di interpretazione perché l’BCNBK restituisce un segnale perfettamente in fase con quello che l’encoder trasmettente produce). La frequenza di lavoro del decodificatore è 1 kHz, perciò è importante accertarsi che l’encoder usato nel trasmettitore abbia l’oscillatore dimensionato per operare anch’esso a 1.000 Hz; altrimenti per il D4ML sarà impossibile “agganciare” gli impulsi in arrivo e decifrarli. Negli encoder HT12E e similari, il clock dell’oscillatore viene stabilito dalla rete R/C applicata al piedino 13.

Ogni volta la decodifica riceve una stringa di dati, essa legge i dieci bit di “indirizzo” quindi li confronta con l’indirizzo locale impostato mediante il dip-switch a 10 vie. Verificato che l’indirizzo locale corrisponde a quello ricevuto dal trasmettitore, il decodificatore va a leggere i bit assegnati ai pulsanti e identifica il pulsante premuto sulla trasmettente. ➤



ELENCO COMPONENTI:

R1÷R4: 1 kohm
R5: 22 kohm
R6: 82 kohm
C1: 4,7 µF 100 VL
elettronico
C2, C4: 100 nF multistrato
C3: 470 µF 25 VL elettrolitico
C5: 470 µF 16 VL elettrolitico
U1: D4ML
U2: 7805
U3: BCNBK
D1÷D5: 1N4007

LD1÷LD4: led 5 mm rosso
RL1÷RL4: relé 12 V
DS1: dip-switch 10 vie

Varie:

- Plug alimentazione
- Morsettiera 3 poli (4 pz.)
- Strip maschio 3 pin
- Jumper
- Spezzone antenna 17 cm.
- Circuito stampato codice S687

Il montaggio del ricevitore non presenta particolari difficoltà. L’antenna può essere realizzata con uno spezzone di filo rigido lungo esattamente 17 centimetri (1/4 d’onda).

Tipicamente, l'indirizzo viene assegnato sulle linee BIT1÷BIT10, mentre i pulsanti sono associati a BIT11 e BIT12. Prestate però attenzione ad un particolare: per scelta del costruttore Aurel, il trasmettitore e il decoder usati in questo progetto codificano i pulsanti su BIT10 e BIT11, quindi gestiscono l'indirizzamento con le linee BIT1÷BIT9 e BIT12. La coppia, se utilizzata insieme, funziona perfettamente; qualora però volesse utilizzare uno dei due insieme a un trasmettitore o ricevitore già in vostro possesso, fate attenzione ai bit utilizzati per codificare i pulsanti: se il TX codifica i 4 pulsanti su BIT10 e BIT11 mentre il ricevitore usa BIT11 e BIT12, dei quattro canali ne funzionano solo due. Diamo per assodato che codifica e decodifica siano programmate in modo identico: riconosciuto il canale, la decodifica attiva le sue due uscite, con le quali comanda una seconda decodifica binaria, di tipo "4 a 1": quest'ultima converte il codice binario a 2 bit relativo ai pulsanti in un codice decimale, quindi attiva l'uscita assegnata al codice in ingresso. Approfondiamo il discorso sul decoder binario, perché comprendere il suo funzionamento permette di capire perché il modulo D4ML funziona in una certa maniera; si è detto che ogni qual volta il decodificatore principale ritiene valida una stringa

ricevuta, esso comunica all'esterno il valore binario del canale cui la trasmissione si riferisce, per mezzo di due uscite logiche.

Ebbene, alla luce di quanto detto nei paragrafi precedenti riguardo alla corrispondenza tra combinazioni logiche e pulsante premuto sul TX, se tale valore ammonta a 00 pone a livello logico alto la sua uscita di peso 0, polarizzando e mandando in saturazione il transistor corrispondente al CH4 (piedino 17) mentre se risulta 01 pone allo stato 1 l'uscita 1 e comanda il transistor relativo al canale 1 (CH1, piedino 14); se la condizione di uscita del decodificatore è 10, si attiva l'uscita del decoder di peso 2, il che manda in saturazione il transistor del terzo canale (CH2, pin 15) e, infine, quando la combinazione ricevuta è 11 il decoder binario porta ad uno logico l'output di peso 3, mandando in saturazione il transistor che comanda l'uscita CH3 (piedino 16).

Le uscite vengono attivate una sola per volta, almeno nella modalità impulsiva tradizionale, nella quale il modulo ibrido comanda il canale corrispondente al pulsante premuto sul trasmettitore, facendolo tornare a riposo al rilascio dello stesso pulsante. Tuttavia abbiamo detto che il dispositivo integra una logica di controllo e temporizzazione delle uscite, che, opportunamente sfruttata, consente modi di funzionamento che permettono l'attivazione simultanea di più canali.

Impostare la modalità

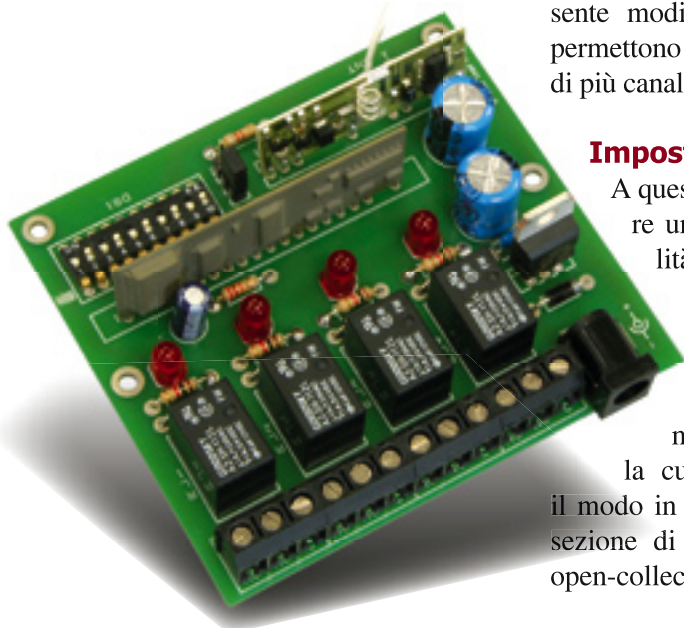
A questo punto è il caso di aprire una parentesi sulle modalità di funzionamento e su come impostarle.

La logica di gestione dell'attività delle uscite fa capo, esternamente, al piedino 12, la cui condizione determina il modo in cui essa interviene sulla sezione di comando dei transistor open-collector di uscita; più precisa-

mente, quando tale pin si trova fisso a livello alto (lo stesso potenziale di alimentazione applicato al piedino 20 rispetto al 18) il D4ML lavora in modalità bistabile, nel senso che ciascuna uscita cambia di stato ogni volta che viene ricevuta una trasmissione valida che la riguarda e mantiene la condizione fino alla ricezione di una ulteriore trasmissione. Più in generale, ogniquale volta si preme uno dei quattro pulsanti del trasmettitore, l'uscita corrispondente si pone nella condizione opposta a quella in cui si trovava in precedenza. In caso di black-out, al ritorno dell'alimentazione le uscite si ripresentano tutte a riposo. Il modo a livello è ottenuto attivando un latch, ossia un quadruplo flip-flop opportunamente connesso che memorizza la condizione delle uscite del decoder binario e la inverte ogni volta che cambia; il latch viene escluso quando il modulo è fatto lavorare ad impulso (modo monostabile).

Vediamo adesso come si attiva la modalità impulsiva, che, è previsto, può essere istantanea o ritardata. Lasciando aperto il piedino 12, i transistor open-collector seguono la condizione logica delle uscite del decoder binario, quindi ripetono (con minimo ritardo, derivante dalle caratteristiche costruttive del trasmettitore e del ricevitore e della decodifica del D4ML) lo stato dei pulsanti del TX; premendo un tasto l'uscita del canale corrispondente sull'RX si attiva e rilasciando la stessa torna a riposo. Invece, collegando tra il pin 12 e massa un bipolo R/C parallelo, si può fare in modo che, al rilascio del pulsante del TX, in modalità impulsiva l'uscita corrispondente torni a riposo con un certo ritardo, che approssimativamente si ricava dal prodotto dei valori di resistenza e condensatore.

Nel nostro ricevitore, il piedino 12 viene gestito mediante il jumper JP1: chiudendo questo verso R5 si ottiene la modalità bistabile, mentre realiz-



zando l'altro ponticello (e aprendo quello verso l'alimentazione) si entra in modalità impulsiva, che può essere ritardata o meno a seconda della presenza della rete R6/C1.

Sulla modalità monostabile con ritardo, va fatta una precisazione: le uscite non possono essere attivate più d'una alla volta, nel senso che se una è ancora attiva (per effetto della temporizzazione introdotta da R6 e C1) quando sul trasmettitore si preme il tasto corrispondente ad un'altra, viene attivata quest'ultima e disattivata istantaneamente la precedente.

Le uscite a relé

Per il nostro circuito, volendo un ricevitore adatto a controllare svariati tipi di utilizzatore abbiamo pensato di sfruttare i transistor open-collector di uscita per pilotare (in modo "sink", cioè ad assorbimento di corrente) le bobine di quattro relé a singolo scambio, di ciascuno dei quali rendiamo disponibili, tramite apposite morsettiere, i contatti comune, normalmente aperto e normalmente chiuso.

Ciascun relé può commutare correnti fino ad 1 ampere in circuiti funzionanti ad un massimo di 120 V (continui o alternati); volendo gestire carichi induttivi, è utile collegare un condensatore da $6,8 \div 10$ nF (in poliestere) e 160 VL tra i contatti usati dello scambio, per limitare la possibilità che in apertura del circuito il carico produca scintille che, a lungo andare, andrebbero a usurare i contatti rendendoli non più affidabili.

Non va infatti scordato che gli elementi dal comportamento induttivo hanno carattere inerziale nei riguardi della corrente, quindi quando si interrompe il circuito che li alimenta tentano di mantenere il flusso di corrente e determinano extratensioni tali da far scoccare l'arco elettrico fra gli estremi dell'interruzione, erodendo i contatti di interruttori e relé.

L'attivazione di ciascun relé è segnalata dall'illuminazione di un led

alimentato dalla stessa uscita che comanda la bobina corrispondente tramite un resistore di limitazione della corrente (R1 per LD1, R2 per LD2, R3 per LD3 ed R4 per LD4).

Per l'alimentazione del ricevitore servono da 12 a 15 V in continua, da applicare ai punti + e - PWR con polarità positiva sull'anodo del D5; qualora si invertisse erroneamente l'alimentazione, tale diodo protegge il circuito perché, essendo polarizzato inversamente, non conduce.

I condensatori C2 e C3 filtrano la componente continua a valle del diodo da disturbi a radiofrequenza ed eventuale residuo d'alternata dell'alimentatore AC/DC; ai loro capi viene prelevata la tensione che serve a far funzionare le bobine dei relé.

I 5 volt stabilizzati occorrenti alla decodifica e al ricevitore ibrido vengono ottenuti mediante un regolatore integrato di tipo 7805, un integrato a

tre piedini che può erogare una corrente di ben 1,5 A, più che sufficiente allo scopo. Poiché la corrente necessaria è molto bassa (una ventina di milliampere) il componente non richiede alcun dissipatore di calore.

Realizzazione pratica

A questo punto, terminato l'esame dello schema elettrico passiamo a dare un'occhiata alla costruzione del ricevitore per il radiocomando (il trasmettitore si acquista già pronto e l'unica operazione da fare su di esso è impostare i 10 bit di codifica); la realizzazione richiede un circuito stampato che può essere facilmente preparato seguendo le tracce lato rame disponibili nell'apposita sezione download del nostro sito Web (www.elettronica.in.it) quindi seguendo il relativo disegno di montaggio. Inciso e forato lo stampato, si parte col montare le resistenze e i ➤

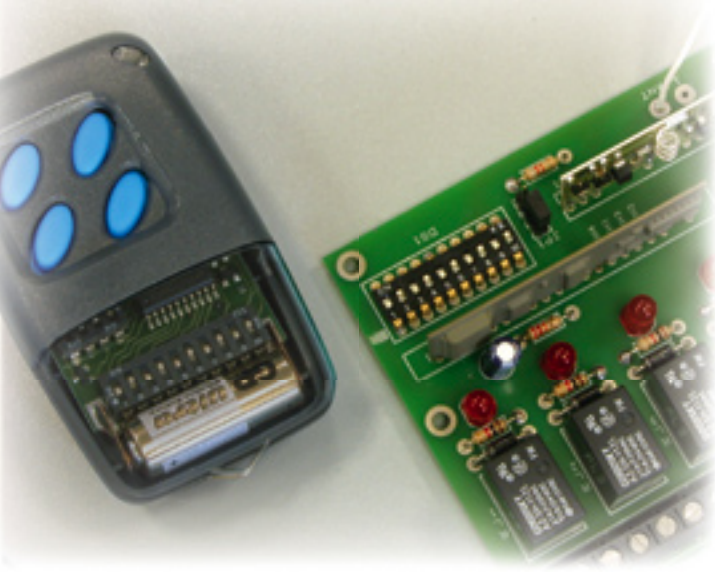
Le modalità di attivazione

Il ricevitore può comandare i relé di uscita affinché operino in maniera impulsiva (con e senza ritardo) o bistabile; l'impostazione corrispondente si effettua mediante il piedino 12 dell'ibrido D4ML, ovvero tramite il jumper JP1, secondo la tabella seguente.

Tabella 1

MODALITA'	FUNZIONE	STATO PIEDINO 12
bistabile	Il relé cambia di stato ogni volta che sul TX si preme il pulsante corrispondente.	Livello alto (+5 V)
impulsiva istantanea	Il relé è normalmente a riposo e viene eccitato dietro comando del pulsante del TX relativo al canale corrispondente; rilasciando il tasto il relé si disattiva subito.	Non collegato
impulsiva con ritardo	Il relé è normalmente a riposo e viene eccitato dietro comando del pulsante del TX relativo al canale corrispondente; rilasciando il tasto il relé ricade con un ritardo determinato all'incirca dal prodotto $R6 \times C1$.	Collegato alla rete R6/C1

Nella modalità impulsiva (monostabile con ritardo) il tempo per il quale l'uscita rimane ancora attiva dopo la sospensione del comando inviato dal trasmettitore dipende dalla costante di tempo RxC ; per ottenere il valore in secondi R deve essere espressa in megaohm e C in microfarad. Il condensatore è bene sia scelto di capacità compresa fra 100 nF e 100 μ F, mentre per R conviene stare fra 10 kohm e 1 Mohm. I valori consigliati dal costruttore sono 82 kohm e 4,7 μ F.



diodi al silicio, badando di rispettare la polarità di questi ultimi e rammentando che il loro terminale di catodo è identificato da una fascetta colorata; subito dopo si possono inserire il dip-switch DS1 (a 10 vie) e la strip a tre punte a passo 2,54 mm che realizzerà l'eventuale ponticello JP1. Allo scopo di correlare l'impostazione della codifica sul ricevitore con quella sul trasmettitore, raccomandiamo di posizionare il dip 1 dalla parte del piedino 1 del D4ML, ovvero in modo che sia elettricamente connesso con quest'ultimo. Ora sistemate i condensatori (stando attenti alla polarità di quelli elettrolitici) e i led, ricordando che per ciascuno il catodo è il terminale che sta dalla parte della smussatura sul contenitore; dopo, potete inserire e saldare i relé, che entreranno in un solo verso, quello giusto. Infilate quindi i moduli ibridi al loro posto, senza preoccuparvi troppo di come orientarli: infatti entreranno ciascu-

no solo in un modo (almeno se avrete preparato la basetta seguendo la nostra traccia); collocate anche il 7805, che dovrà stare posizionato con il lato metallico rivolto al condensatore C2. Allo scopo di agevolare le connessioni con gli scambi del relé e con l'alimentatore, prevedete idonee

morsettiere da circuito stampato a passo 5 mm, da infilare e saldare nelle rispettive piazzole. Per completare l'unità ricevente, tagliate uno spezzone di filo di rame rigido (se è smaltato raschiatene un'estremità in modo da poterlo saldare, altrimenti lo stagno non attacca) lungo 17 centimetri e infilatelo nel foro ANT della basetta, saldandolo alla relativa piazzola. Bene, adesso il ricevitore è pronto, dato che non necessita di alcuna taratura preliminare; per collaudare il radiocomando basta dare alla ricevente una tensione continua di 12÷15 volt, applicandola con il positivo al punto +V ed il negativo a massa (-V) dello stampato, prelevandola da un alimentatore o una batteria capaci di fornire almeno 200 milliamper. Prima di lavorare è indispensabile impostare i dip-switch per avere uguale codifica su trasmettitore e ricevitore: allo scopo basta impostare i bit del D4ML come quelli dell'encoder posto sul

trasmettitore, ovvero, ad esempio, se nel ricevitore il primo dip del DS1 è chiuso e tutti i restanti sono aperti, bisogna fare lo stesso sul TX, cioè chiudere (ON) il solo dip 1 del DS1. Fatto ciò, non resta che verificare se tutto è a posto: basta premere un tasto del trasmettitore e verificare che sul ricevitore scatti il led corrispondente, secondo la modalità definita dal JP1 (si veda, a riguardo, l'apposita tabella). Insieme al relé, deve attivarsi (illuminarsi) il led corrispondente. Accertato che tutto è a posto, conviene staccare l'alimentazione del ricevitore e trovare una sistemazione per lo stampato: l'ideale è un contenitore plastico, perché lascia passare le onde radio in arrivo dalla trasmittente; optando per un involucro metallico, bisogna collocare l'antenna all'esterno, connettendola mediante uno spezzone di cavo coassiale. A proposito di antenna, allo scopo di aumentare la portata del sistema, si può sostituire lo spezzone di filo con uno stilo o una ground-plane accordati a 433,92 MHz. Un'ultima nota riguarda i relé utilizzati, i cui contatti operano a bassa tensione, per cui non sono idonei a comandare carichi a 230 Vac. In questo caso occorre utilizzare dei relé aggiuntivi i cui contatti possano commutare la tensione di rete; la bobina va pilotata con il contatto normalmente aperto dei relé RL1÷RL4; suggeriamo di utilizzare dei relé a 12 V per poter sfruttare la stessa tensione di alimentazione del circuito.

per il

MATERIALE

Il progetto descritto in queste pagine è disponibile in scatola di montaggio. Il trasmettitore (cod. TX12E-4C-SAW 433) è disponibile già montato e collaudato al costo di 28,00 Euro mentre il kit del ricevitore a 4 canali (cod. FT687K) costa 40,00 Euro. La scatola di montaggio comprende tutti i componenti elettronici, i relé, il ricevitore BCNBK, il decoder D4ML e la basetta forata e serigrafata. I prezzi si intendono IVA compresa.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-778112 ~ <http://www.futuranet.it>



Liquid Crystal Display dalla teoria alla pratica

Impariamo a conoscere e utilizzare uno dei componenti elettronici più interessanti: il display LCD. In questa puntata scopriamo tutti i segreti del controller grafico Samsung KS0108B, componente poco conosciuto ma largamente utilizzato in molti display grafici.

A CURA DI *Matteo Destro*

5

In questa puntata ci occupiamo del display grafico 64128ABCB e soprattutto del suo controller, denominato KS0108B e prodotto da Samsung. Dopo una panoramica sulle potenzialità di questo chip, ne analizzeremo i comandi disponibili e termineremo la puntata con lo studio di alcune routine create appositamente per questo controller.

Il controller grafico KS0108

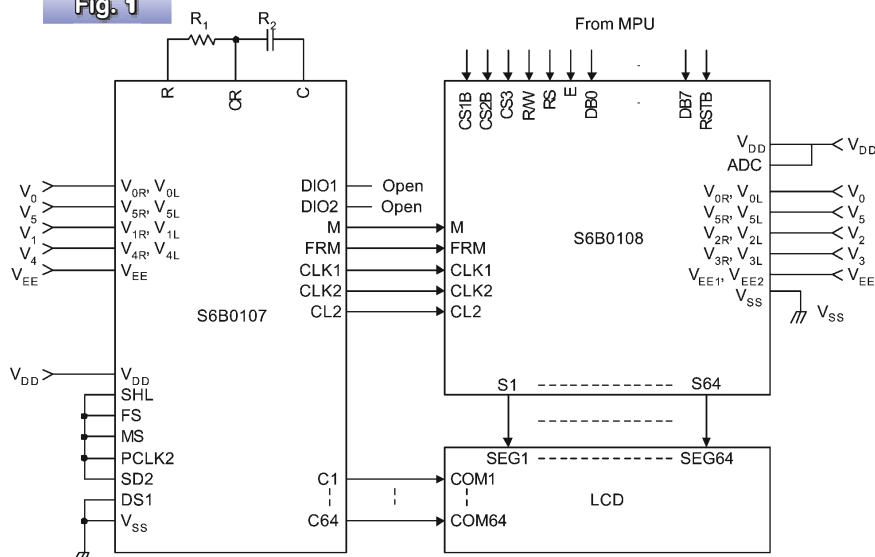
È un driver per display LCD realizzato in tecnologia LSI (Large Scale Integration = integrazione a larga scala), tecnologia che ha permesso di integrare in un unico dispositivo sia i circuiti elettronici necessari per pilotare i pixel del vetrino dell'LCD, sia la logica di controllo e interfacciamento con un microcontrollore, nonché la memoria RAM necessaria per immagazzinare le informa-

zioni (immagini, font o altro) da visualizzare sul display. La dimensione della RAM interna è di 512 byte, taglio sufficiente per gestire display dalle dimensioni massime di 64x64 pixel. Poiché il display di cui parliamo in questo articolo è un 128x64 pixel, ne consegue che esso necessita di due controller: il primo si occupa della gestione della prima metà del display (64x64 pixel) mentre il secondo controlla la restante metà.

I controller della serie KS0108B sono sempre associati ad un secondo integrato, il KS0107B, collegato al primo come indicato in Figura 1.

Come si può notare nelle illustrazioni, il controller KS0108B si occupa della gestione dei segmenti (SEG1... SEG64) mentre l'integrato KS0107B si occupa della gestione delle linee comuni (COM1... COM64). I due integrati si scambiano le informazioni necessarie per la visualizzazione ➤

Fig. 1



Edge, attivazione sul fronte di discesa); successivamente viene trasferito nella locazione di memoria RAM associata. L'operazione di scrittura in RAM è interna al controller KS0108B.

Il terzo blocco è l'“**Output Register**”, che memorizza temporaneamente il dato letto dalla RAM oppure lo stato del controller il quale può avvisare il microprocessore di essere “occupato”, ad esempio in un processo di scrittura. Per leggere un

delle immagini sul display tramite alcuni segnali di controllo.

Analizziamo lo schema a blocchi del KS0108B (Figura 2) iniziando dal blocco “**I/O Buffer**” con cui si interagisce con il driver. I segnali CS1B e CS2B sono collegati internamente a massa mentre il pin CS3 viene gestito dal microcontrollore. Il controller è disabilitato fino a quando il pin CS3 è a livello logico basso; viceversa un livello logico alto ne comporta l'abilitazione. Oltre ai pin già descritti, convergono in questo blocco funzionale anche i rimanenti pin di controllo, necessari per distinguere le istruzioni dai dati (RS) e le operazioni di lettura da quelle di scrittura (R/W), nonché il segnale per sincronizzare l'invio di dati e istruzioni (E) e il segnale di reset (RSTB) con cui riportare il chip alla condizione di partenza.

Il secondo blocco è l'“**Input Register**”, che memorizza temporaneamente i dati ricevuti dal micro-controllore prima di scriverli nella memoria RAM. Per accedere in scrittura all'Input Register è necessario porre R/W a livello logico basso mentre RS deve essere a livello logico alto. Il dato viene copiato nel registro di input quando il segnale E passa da livello logico alto a livello logico basso (Falling

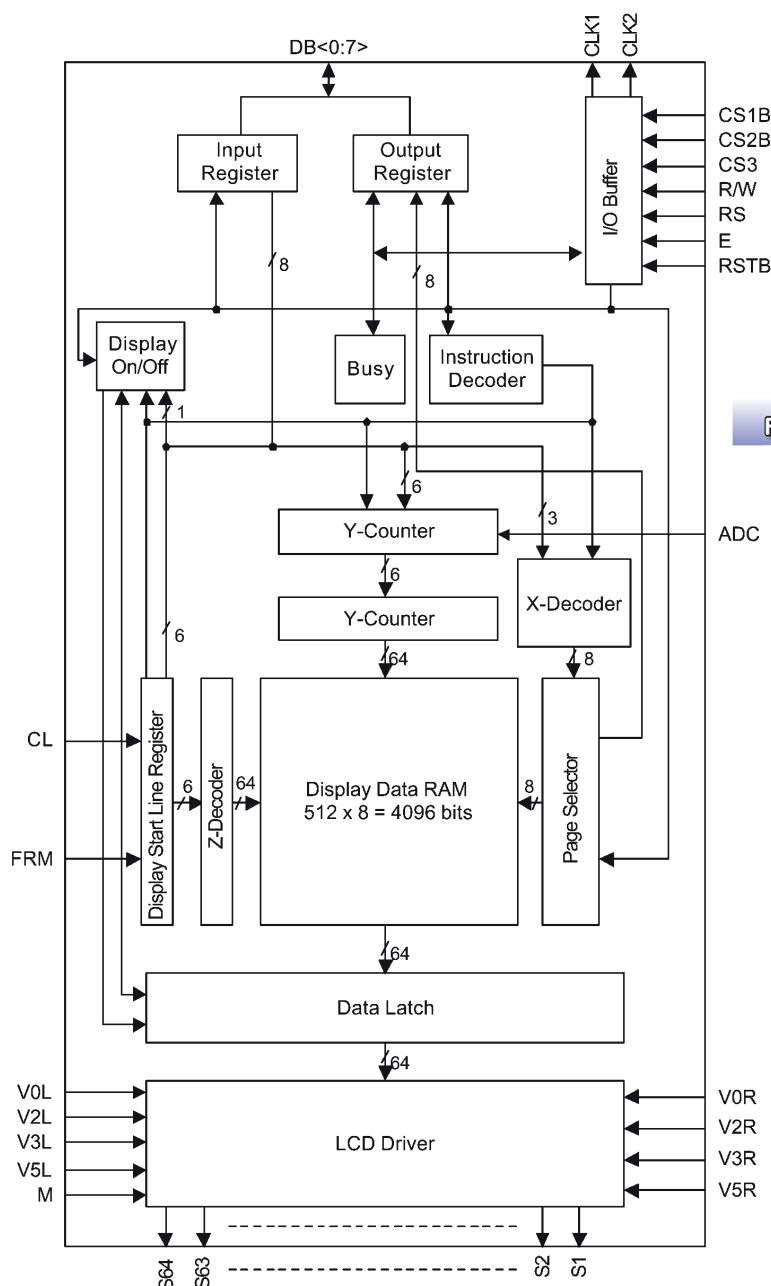


Fig. 2

dato dalla RAM è necessario impostare sia R/W che RS a livello logico alto, mentre per leggere lo stato del controller è necessario impostare R/W a livello logico alto e RS a livello logico basso. Nel caso si voglia leggere un dato dalla memoria RAM è necessario eseguire una doppia lettura: la prima serve per spostare il dato dalla RAM all'**Output Register** (Lettura Dummy) e la seconda per trasferire il dato dall'Output Register al bus dati, rendendolo pertanto disponibile al microcontrollore. Se invece si deve estrapolare lo stato del display è sufficiente una sola lettura.

Il quarto blocco, "**busy**", serve per indicare se il controller è occupato oppure se è disponibile ad accettare un dato o una istruzione. Quando il Busy Flag è alto significa che il controller è occupato, in caso contrario il controller è disponibile a ricevere dati e istruzioni. Questa informazione si trova sul bit più significativo del bus dati cioè sul DB7.

Segue l'**"Instruction Decoder"** il quale, come suggerisce il nome, serve per decodificare i comandi inviati dal micro-controllore al KS0108B. Il sesto blocco è definito come **"X Decode o X Page Register"** e si occupa di selezionare la pagina di RAM interna al controller. Il controller non dispone di meccanismi di autoincremento della pagina, perciò è necessario impostare di volta in volta la pagina di RAM su cui si vuole scrivere prima di inviare i dati da visualizzare. Esiste quindi un apposito comando per impostare la pagina RAM da utilizzare.

Il settimo blocco è l'**"Y Counter"** o **"Y Address Counter"** e, in modo simile al precedente blocco, si occupa di indirizzare la memoria RAM interna al controller. L'indirizzo viene impostato tramite comando, ma in questo caso viene incrementato automaticamente di una unità ad ogni lettura/scrittura sulla memoria RAM del controller.

Quindi ne consegue che, per scrivere in una data posizione sul display, è necessario impostare in

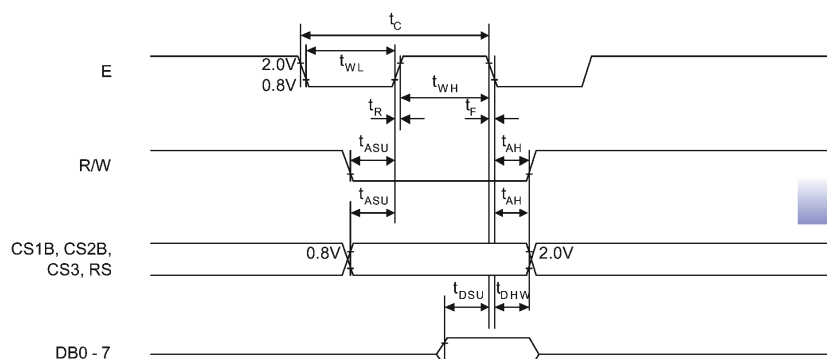


Figura 3: MPU Write Timing

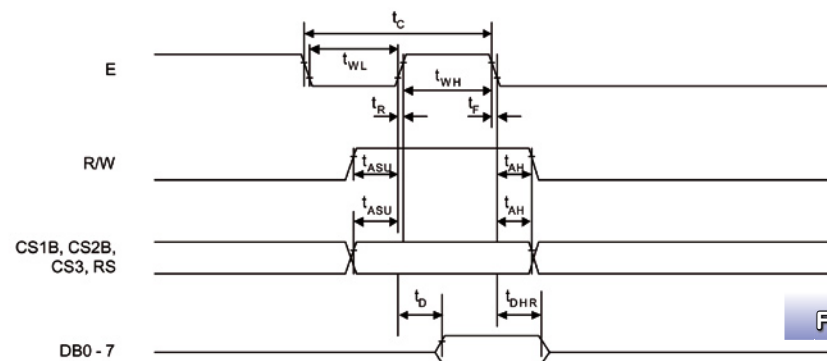


Figura 4: MPU Read Timing

modo corretto prima la pagina e poi l'indirizzo della coordinata Y.

L'ottavo blocco è il **"Display Data RAM"** il quale si occupa di memorizzare i dati da visualizzare sul display (un bit per ogni pixel). Per accendere un pixel è necessario scrivere un livello logico "1" nel rispettivo bit. Un livello logico "0", invece, spegne il pixel.

L'ultimo blocco è il **"Display Start Line Register"** che viene utilizzato per effettuare lo "spostamento" dell'immagine visualizzata sul display. Infatti l'immagine sarà più o meno spostata rispetto all'asse Y a seconda del valore impostato in questo registro.

Tutti i blocchi funzionali descritti vengono governati per mezzo delle istruzioni di cui il controller grafico dispone, istruzioni che verranno descritte a breve. Prima di iniziare a parlarne, però, è conveniente studiare i diagrammi temporali: affinché il display funzioni correttamente, è infatti necessario che il microcontrollore esterno, ad esempio un PIC 18F4620, rispetti tali tempistiche.

In figura 3 è visibile il diagramma temporale relativo ad una operazione di scrittura in RAM. Va tenuto presente che i pin CS1B e CS2B, come già anticipato, sono collegati direttamente a massa sul PCB del controller grafico.

È fondamentale rispettare il tempo " t_c " il quale ➤

Tabella 1

Descrizione	Simbolo	Min	Typ	Max	Unità di misura
Tempo di ciclo segnale E	t_c	1000	-	-	nSec
Larghezza segnale E livello alto	t_{WH}	450	-	-	nSec
Larghezza segnale E livello basso	t_{WL}	450	-	-	nSec
Tempo di salita segnale E	t_R	-	-	25	nSec
Tempo di discesa segnale E	t_F	-	-	25	nSec
Tempo di Set-Up indirizzo	t_{ASU}	140	-	-	nSec
Tempo di Hold time indirizzo	t_{AH}	10	-	-	nSec
Tempo di Set-Up dato	t_{DSU}	200	-	-	nSec
Tempo di ritardo dato	t_D	-	-	320	nSec
Tempo di Hold time (Scrittura)	t_{DHW}	10	-	-	nSec
Tempo di Hold time (Lettura)	t_{DHR}	20	-	-	nSec

non deve mai essere inferiore a 1000 nsec. Il dato viene scritto in memoria quando il segnale di Enable “E” passa dal livello logico alto al livello logico basso alla fine del tempo di ciclo “ t_c ”. Il diagramma temporale per un’operazione di lettura (Figura 4), invece, differisce leggermente solo per lo stato del segnale R/W.

Anche in questo caso il dato è disponibile sul bus dopo che il segnale di Enable “E” è passato da livello logico alto a livello logico basso alla

fine del tempo di ciclo “ t_c ”. In Tabella 1 vengono riportati tutti i tempi da rispettare per una corretta gestione del display.

Il set di comandi

Descriviamo ora i comandi messi a disposizione dal controller KS0108B e verifichiamo la loro interazione con i blocchi funzionali descritti precedentemente. Come si può osservare in Tabella 2, il numero dei comandi disponibili è molto esiguo ma comunque sufficiente per gestire il display. Dal numero limitato di comandi emerge una prima considerazione: i display grafici basati su questo controllore devono essere utilizzati nei progetti che non richiedono una gestione troppo spinta della grafica.

Tabella 3

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	1	1	1	0

Tabella 2

Comando	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Funzione
Display On/Off	L	L	L	L	H	H	H	H	H	L/H	On/Off accende e spegne il display LCD. L = OFF; H = ON.
Set Address (Y Address)	L	L	L	H	Indirizzo Y (0÷63)						Imposta indirizzo Y nel registro Address Counter.
Set Page (x Address =	L	L	H	L	H	H	H	Pagina X (0÷7)			Imposta pagina X nel registro Address Register.
Display start line	L	L	H	H	Inizio linea display (0÷63)						Indica da quale linea iniziare a visualizzare l'immagine sul display. Esegue uno scroll rispetto all'asse Y.
Status Read	L	H	Busy	L	On/Off	Rst	L	L	L	L	Stati letti: Busy: L: Pronto H: Occupato On/Off: L: Display ON H: Display OFF Reset: L: Modalità normale H: Modalità reset
Write Display	H	L	Scrivi Dato								Scriva il dato nella locazione di RAM selezionata. Dopo la scrittura, l'indirizzo Y viene automaticamente incrementato di 1.
Read Display	H	H	Leggi Dato								Legge il dato dalla locazione di RAM selezionata e lo rende disponibile sul bus dati.

Display ON/OFF

Dettagliato in Tabella 3, in questo comando è possibile impostare solo il bit meno significativo “D”. Se D = 0 l’immagine non viene visualizzata mentre se D = 1 essa viene visualizzata sul display. Questo comando non influenza il contenuto della memoria RAM, cioè se si imposta D = 0 l’immagine memorizzata nella memoria RAM rimane integra e non subisce variazioni di nessun genere. I restanti bit, da DB1 a DB7 sono fissi al valore visualizzato nella tabella e non devono essere modificati. Entrambi i segnali RS e R/W sono a livello logico basso perché si tratta di un comando.

Set Address (Y Address)

Questo comando serve per impostare l’indirizzo di memoria RAM in cui si vuole leggere o scrivere. È abbinato all’indirizzo di pagina in quanto, a seconda della pagina selezionata, si va ad agire su locazioni di memoria differenti. Come si osserva dalla Tabella 4 possiamo impostare sei degli otto

Tabella 4

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

bit a disposizione per un totale di 64 locazioni di memoria. Durante le operazioni di scrittura/lettura su RAM, l’indirizzo “Y Address” viene automaticamente incrementato di una unità: è sufficiente quindi inviare una sola volta il comando di indirizzamento dopodiché saranno le istruzioni di read e write ad occuparsi dell’auto incremento e quindi della lettura o scrittura alla locazione successiva in RAM.

Set Page (X Address)

Questo comando serve per impostare la pagina di RAM su cui si vuole lavorare (Tabella 5).

Tabella 5

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	1	1	1	AC2	AC1	AC0

La memoria RAM è suddivisa in otto pagine e ogni pagina è formata da 64 byte (vedere anche “Set Address (Y Address)”, comando interagente con quello descritto qui). La Tabella 6 che segue mette in evidenza quanto detto: per ogni pagina vengono associati 64 byte (Seg.1... Seg.64). La prima pagina si trova all’indirizzo 0xB8 mentre l’ultima è localizzata a 0xBF.

Quindi se volessimo scrivere nella locazione 5 della pagina 2 dovremmo impostare “X Address” a 0xB9 e “Y Address” a 0x45.

Tabella 6

		Y Address							
X Address	Dato	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	...	...	Seg. 63
B8	D0	0	1	1	1	0	...	...	...
	D1	1	0	0	0	1	...	...	...
	D2	1	0	0	0	0	...	...	...
	D3	0	1	1	1	0	...	...	...
	D4	0	0	0	0	1	...	...	...
	D5	1	0	0	0	1	...	...	...
	D6	0	1	1	1	0	...	...	...
	D7	0	0	0	0	0	...	...	...
B9	D0	X	X	X	X	X	...	...	...
	D1	X	X	X	X	X	...	...	...
	D2	X	X	X	X	X	...	...	...
	D3	X	X	X	X	X	...	...	...
	D4	X	X	X	X	X	...	...	...
	D5	X	X	X	X	X	...	...	...
	D6	X	X	X	X	X	...	...	...
	D7	X	X	X	X	X	...	...	...
...	...	...	...	...	...	...	...	...	...
BF	D0	X	X	X	X	X	...	...	...
	D1	X	X	X	X	X	...	...	...
	D2	X	X	X	X	X	...	...	...
	D3	X	X	X	X	X	...	...	...
	D4	X	X	X	X	X	...	...	...
	D5	X	X	X	X	X	...	...	...
	D6	X	X	X	X	X	...	...	...
	D7	X	X	X	X	X	...	...	...

Display Start Line (Z Address)

Serve per far scorrere il display rispetto all’asse Y a seconda del valore di “Z Address” (Tabella 7).

Tabella 7

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	1	AC5	AC4	AC3	AC2	AC1	AC0

Per rendere più comprensibile questo comando è conveniente fare riferimento alle immagini di esempio visualizzate nelle Figure 5a, 5b e 5c. L’immagine “5a” ha il registro “Z Address” impostato a 0x00 e quindi non presenta alcuno spostamento rispetto all’asse Y. L’immagine “5b” ha subito uno “scorrimento”: il registro “Z Address” è impostato a 0x0A: ne deriva che la



parte superiore dell'immagine ricompare nella parte inferiore del display.

L'effetto dello spostamento è tanto più evidente quanto maggiore è il valore del registro "**Z Address**": nella figura "**5c**" è stato impostato a 0x0F.

Status Read

Questo comando, mappato come indicato in Tabella 8, serve per leggere lo stato del controller KS0108B.

Tabella 8

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BUSY	0	ON/OFF	RST	0	0	0	0

Durante l'operazione di lettura vengono rese disponibili sul bus dati le seguenti informazioni:

- **Busy**: Indica se il controller è occupato nell'esecuzione di comandi precedentemente ricevuti. Quindi se leggiamo che il bit di BUSY è uguale a "1" logico significa che il controller è occupato e non può accettare altre istruzioni o dati. Invece se BUSY è uguale a "0" logico significa che il controller è disponibile e quindi pronto a ricevere dati o istruzioni.
- **ON/OFF**: Indica se il display è attivo o disattivo. Se ON/OFF è uguale a "1" logico significa che il display è OFF e quindi l'immagine contenuta nella memoria RAM non è visibile. Se invece ON/OFF è a "0" logico significa che il display è acceso e quindi l'immagine presente in RAM è visibile sul display. Quando il display è nello stato di OFF, i dati in RAM vengono mantenuti.
- **RST**: Indica se il controller è nella fase di inizializzazione. Se RST è uguale a "1" logico significa che il sistema sta entrando in funzione e quindi nessuna istruzione può essere eseguita tranne la lettura dello stato. Se invece RST è uguale a "0" logico significa che l'inizializzazione è stata conclusa e il sistema è pronto a ricevere comandi e dati.

Write display Data

Il microprocessore utilizza questo comando (mappato come in Tabella 9) per inviare al display i dati

da scrivere nella memoria RAM: deve essere preceduto almeno una volta, dai comandi "**Set Address (Y Address)**" e "**Set Page (X Address)**".

Tabella 9

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Il registro Y Address viene automaticamente incrementato di una unità ad ogni scrittura in RAM, puntando alla locazione di memoria RAM successiva. Se si deve cambiare pagina è necessario inviare nuovamente i comandi "**Set Address (Y Address)**" e "**Set Page (X Address)**". Si osservi che in questo caso il segnale RS è a livello logico alto individuando così l'invio di un dato.

Read display Data

Questo comando serve per leggere i dati dalla memoria RAM ed utilizza i bit in Tabella 10.

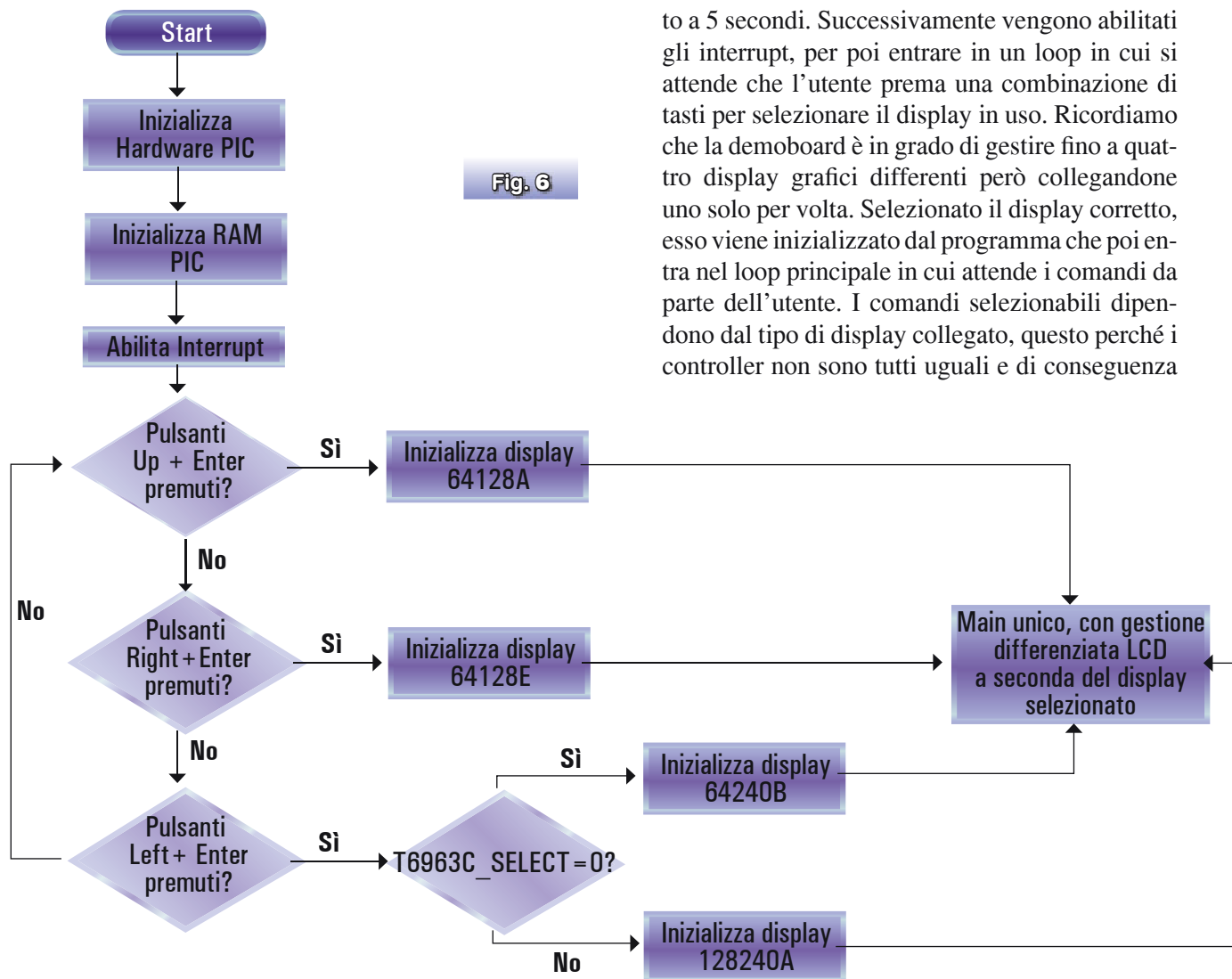
Tabella 10

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Quando si esegue una lettura della memoria RAM è necessario eseguire una doppia lettura. La prima viene detta "lettura Dummy" e serve per trasferire il dato voluto dalla memoria RAM all'output register, mentre la seconda lettura preleva il dato. Ad ogni lettura il registro Y Address viene automaticamente incrementato di una unità puntando alla successiva locazione di memoria RAM. Come già visto, dovendo cambiare pagina è necessario rinviare i comandi "**Set Address (Y Address)**" e "**Set Page (X Address)**".

Firmware per il controller KS0108B

Il firmware necessario per la corretta gestione del controller KS0108B è stato realizzato in linguaggio ANSI C per PIC 18F4620. Prima di iniziare a commentare le routine è necessario avere una visione panoramica di come vengono gestiti i quattro display utilizzando un solo micro-controllore e quindi un solo firmware. Per agevolare la descrizione utilizzeremo lo schema a blocchi di Figura 6.



to a 5 secondi. Successivamente vengono abilitati gli interrupt, per poi entrare in un loop in cui si attende che l'utente prema una combinazione di tasti per selezionare il display in uso. Ricordiamo che la demoboard è in grado di gestire fino a quattro display grafici differenti però collegandone uno solo per volta. Selezionato il display corretto, esso viene inizializzato dal programma che poi entra nel loop principale in cui attende i comandi da parte dell'utente. I comandi selezionabili dipendono dal tipo di display collegato, questo perché i controller non sono tutti uguali e di conseguenza

Per prima cosa è necessario inizializzare le periferiche hardware interne al PIC18F4620, che nel nostro caso sono le seguenti:

- I²C per gestire la EEPROM 24FC1025 esterna al micro;
- UART tramite la quale la demoboard colloquia con la RS232 del computer;
- Timer interni;
- Porte di I/O;
- Registri di controllo e gestione degli interrupt.

Conclusa l'inizializzazione delle periferiche hardware è necessario inizializzare la memoria RAM. Per prima cosa è conveniente impostare tutte le locazioni di memoria a un valore noto (nel nostro caso 0x00). A questo punto si devono inizializzare le variabili che devono avere un valore diverso da zero: vediamo quali sono. La prima è il "timeout" della trasmissione seriale RS232, che va imposta-

ognuno di essi ha delle peculiarità che lo distinguono dagli altri. Per esempio, i display 64240B e 128240A hanno lo stesso controller ma, vista la differente dimensione in pixel, devono essere inizializzati in modi differenti. Per questione di semplicità nel codice, abbiamo deciso di differenziare i due display tramite la "compilazione condizionale": se la costante "T6963C_SELECT", contenuta nel file header "t6963c.h", è impostata a 0x00, vengono usati i parametri per inizializzare il display 64240B, mentre se tale costante è inizializzata a 0x01 si utilizzano i parametri per il display 128240A. Quindi l'utente deve ricordarsi di impostare questa costante prima di compilare il progetto e programmare il PIC18F4620.

È stato realizzato un file contenente tutte le routine necessarie alla gestione per ogni controller grafico quindi, una volta avviato l'ambiente di sviluppo MPLAB IDE e caricato il progetto, deve apparire l'elenco di file, visibile in figura 7, in cui i file con estensione ".c" rappresentano il sorgente speci-

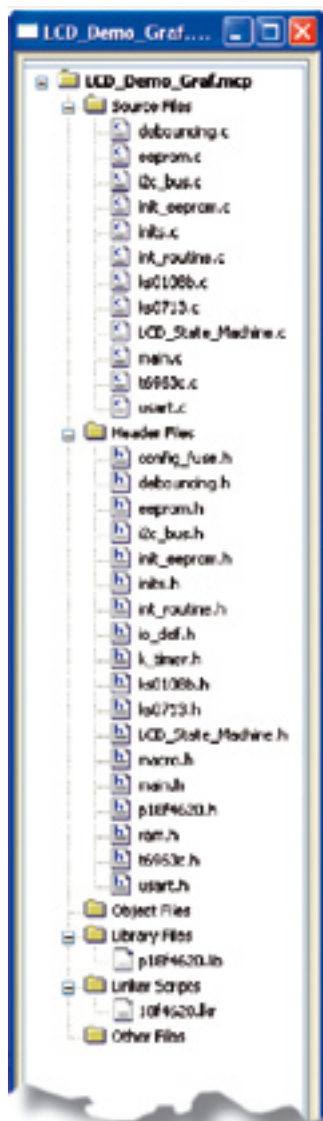


Fig. 7

fico per ogni display e routine, mentre i file con estensione “.h” rappresentano gli header.

I file più importanti sono:

- “**ks0108b.c**” e relativo file “**ks0108b.h**”. Routine di inizializzazione e gestione controller KS0108B con relativi parametri.
- “**ks0713.c**” e relativo file “**ks0713.h**”. Routine di inizializzazione e gestione controller KS0713 con relativi parametri.
- “**t6963.c**” e relativo “**t6963.h**”. Routine di inizializzazione e gestione controller KS0713 con relativi parametri.

Oltre ai file contenenti le routine di gestione dei controller, è stato realizzato un file contenente la macchina a stati di gestione dei display, che esamina e gestisce i comandi impartiti dall'utente ed interviene di conseguenza sul display connesso alla demoboard. Il file contenente il codice di

gestione della macchina a stati per i display LCD si chiama “**LCD_State_Machine.c**” e il relativo file header “**LCD_State_Machine.h**”.

Di seguito riportiamo il codice relativo al main menu, identico per tutti i display utilizzati (Listato 1).

Listato 1

```
while (TRUE)
{
    ClrWdt();
    if ((GenFlags.TX_RS232 != 1) && (GenFlags.RX_RS232 != 1))
    {
        Debounce_PORTA();
        Process(I2C_State);
        Process(LCD_State);
    }
    else
    {
        // Trasmissione seriale in corso
        if (GenFlags.RX_RS232 == 1)
        {
            Check_RX_Data();
        }
    }
}
```

Come si può osservare è stato realizzato un loop in cui vengono eseguite in modo ciclico alcune chiamate a funzioni tra cui l'esecuzione dei processi di due macchine a stati.

La prima riguarda la gestione del protocollo di comunicazione I²C mentre la seconda è coinvolta nella gestione dei display. Viene anche richiamata la routine di gestione e controllo dei pulsanti connessi alla porta A del PIC18F4620.

Quando si attiva la comunicazione seriale le tre chiamate descritte vengono ignorate e si processa solo la chiamata a funzione “**check_RX_Data()**”. Dopo questa breve panoramica del firmware, si può passare alla descrizione delle routine specifiche per il KS0108B.

Per prima cosa è necessario inizializzare il controller: lo schema a blocchi esposto in Figura 8 riassume le operazioni da eseguire durante il processo di init.

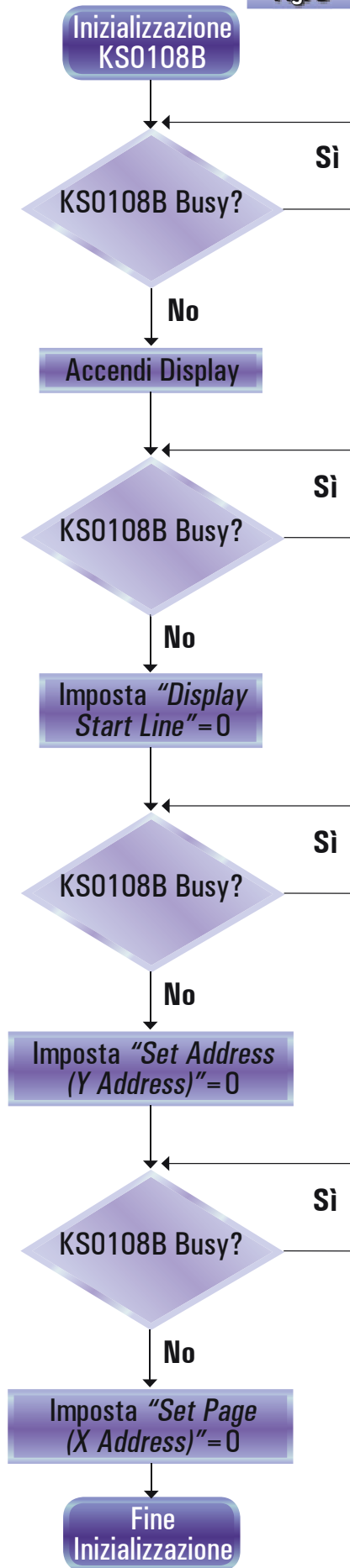
Prima di inviare un comando è necessario testare se il controller è libero. Quando il display è pronto si possono inizializzare i seguenti parametri:

- Display ON
- Display Start line
- Set address (Y Address)
- Set Page (X Address)

Come detto precedentemente il display 64128A ha una risoluzione di 128x64 pixel e quindi necessita di due controller KS0108B: di conseguenza è necessario inizializzarli entrambi.

La routine che si occupa della inizializzazione >

Fig. 8



Listato 2

```

void Init_Procedure_KS0108B(uchar Chipset)
{
    if (Chipset != 1)
    {
        CS2_LCD_M(HIGH);
    }
    else
    {
        CS1_LCD_M(HIGH);
    }

    Test_KS0108B_Busy();
    Set_ON_KS0108B();

    Test_KS0108B_Busy();
    KS0108B_Start_Line = 0;
    Set_Start_Line_KS0108B();

    Test_KS0108B_Busy();
    KS0108B_Y_Address = 0;
    Set_Address_KS0108B();

    Test_KS0108B_Busy();
    KS0108B_Page = 0;
    Set_Page_KS0108B();

    CS1_LCD_M(LOW);
    CS2_LCD_M(LOW);
}
  
```

Discriminazione ed abilitazione del controller. Può essere abilitato un solo controller alla volta.

Display ON.

Impostazione start line. Valore di default "0".

Impostazione Y Address. Valore di default "0".

Impostazione X Address (Page). Valore di default "0".

Legenda:

- In verde vengono evidenziate le funzioni e i relativi parametri
- In rosso vengono evidenziate le chiamate a funzione e in blu i rispettivi parametri
- In blu vengono evidenziate le istruzioni
- In azzurro vengono evidenziati i commenti

Per meglio comprendere meglio i listati, fare sempre riferimento alla legenda qui a fianco.

è visibile nel Listato 2. Notate che è necessario passare alla routine un parametro numerico a 8 bit senza segno, parametro che identifica quale controller è da inizializzare: il valore "1" identifica il controller numero 1 abbinato al segnale di controllo e abilitazione CS1; analogamente il valore "2" identifica il secondo controller, abbinato al segnale di controllo CS2.

Ciò significa che questa routine deve essere richiamata due volte durante l'inizializzazione. Per questione di semplicità, le routine appena descritte sono state raggruppate in una sola più grande, che controlla l'inizializzazione del display 64128A e che include altre due "sub-routine" necessarie per inizializzare la memoria RAM dei due controller, in modo da fissare il contenuto della stessa a dei valori noti. Si è scelto di inizializzare la RAM a 0x00 per avere il display completamente bianco (Listato 3).

Come già detto in precedenza, prima di inviare un comando o un >

Listato 3

```

void Init_KS0108B(void)
{
    Set_Pin_KS0108B();

    //-----
    // Impostazione KS0108B (CS1)
    Init_Procedure_KS0108B(KS0108B_CS1);
    //-----
    // Impostazione KS0108B (CS2)
    Init_Procedure_KS0108B(KS0108B_CS2);
    //-----
    Clear_RAM_KS0108B(KS0108B_CS1);
    Clear_RAM_KS0108B(KS0108B_CS2);
}
  
```

Inizializza controller 1.

Inizializza controller 2.

Inizializza memoria RAM del controller 1 e 2.

dato al controller, è necessario verificare se quest'ultimo è libero e quindi pronto a ricevere dati e comandi oppure se è occupato e quindi impossibilitato a instaurare una comunicazione con il micro-controllore. Cominciamo con l'analizzare la routine di test della condizione di "Busy" del controller (Listato 4).

Listato 4

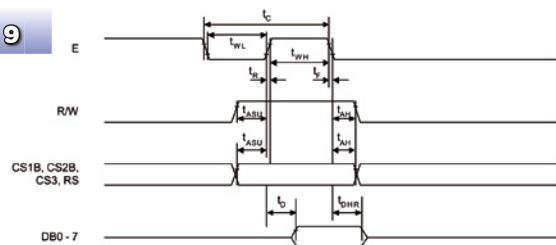
```
void Test_KS0108_Busy(void)
{
    TRISD = 0b11111111;

    Status_Read:
        E_LCD_M(LOW);
        Nop();
        Nop();
        RW_LCD_M(HIGH);
        RS_LCD_M(LOW);
        Nop();
        Nop();
        E_LCD_M(HIGH);
        Nop();
        Nop();
        KS0108B_Status = PORTD & 0x80;
        Nop();
        Nop();
        E_LCD_M(LOW);
        RW_LCD_M(LOW);

        if (KS0108B_Status != 0)
        {
            goto Status_Read;
        }

    TRISD = 0b00000000;
}
```

Fig. 9



Osservate che il codice per la lettura dello stato del controller segue l'andamento del diagramma temporale esposto in Figura 9. Eventualmente, rivedete la tabella 1. Per prima cosa si deve inizializzare la Porta D del PIC come ingresso, e successivamente "muovere" i segnali di controllo in modo da eseguire una lettura dello stato del controller. Il valore letto dalla Porta D viene mascherato con il valore 0x80 e se il risultato è diverso dal valore logico "0" si rimane in loop in attesa che il controller si liberi dalla condizio-

Listato 5

```
void Set_ON_KS0108B(void)
{
    Write_Comm_KS0108B(KS0108B_DISPLAY_ON);
}
```

Invia comando al controller KS0108B.

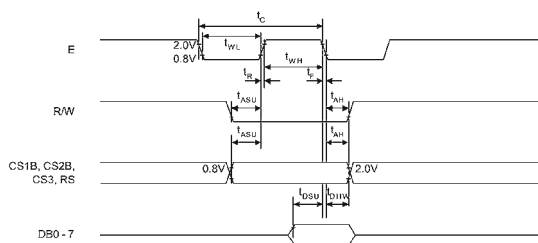


Fig. 10

ne di Busy. Una volta verificata la condizione di "libero" la Porta D viene reimpostata come uscita. Passiamo ora alla descrizione delle routine che si occupano di inviare i comandi o i dati al controller presente sul display grafico 64128A. Cominciamo con la descrizione del comando di accensione del display LCD (Listato 5). Esso è composto da una sola istruzione, che chiama una funzione a cui va passato il codice del comando da inviare al controller grafico. Tale codice comando è identificato dalla costante "KS0108B_DISPLAY_ON" e in questo caso il valore esadecimale da inviare è 0x3F. Per spegnere il display LCD bisogna inviare il codice 0x3E identificato dalla costante "KS0108B_DISPLAY_OFF".

Tutte le costanti utilizzate nel firmware sono memorizzate nel file header "ks0108b.h".

La funzione richiamata serve per gestire i segnali di controllo del KS0108B durante l'invio dei comandi. L'andamento di questi segnali rispecchia il diagramma temporale già visto (riportato in Fig. 9) ed è controllato dal codice visibile nel Listato 6. La prima parte della funzione (1) si occupa di impostare i segnali di R/W, RS nonché il segnale di enable (E) mentre la seconda parte (2) conclude la gestione di questi segnali di controllo: in particolare il segnale di Enable viene portato da valore logico alto a valore logico basso abilitando così il controller ad accettare il comando presente sul bus dati (Porta D del PIC18F4620).

Prima della conclusione della seconda parte della sub-routine è necessario riportare il segnale di Enable a livello logico alto, come da diagramma temporale. Anche i due segnali R/W e RS vengono portati a livello logico alto.

Vengono introdotte delle istruzioni "Nop" tra le varie istruzioni di controllo: esse introducono dei ritardi necessari per rispettare il diagramma temporale. Ogni istruzione "Nop" introduce 100nSec di ritardo in quanto il microcontrollore lavora a una frequenza di 10 MHz e quin-

Listato 6

```
void Write_Comm_KS0108B(uchar KS0108B_Data)
{
    E_LCD_M(LOW);
    Nop();
    Nop();
    RW_LCD_M(LOW);
    RS_LCD_M(LOW);
    Nop();
    Nop();
    E_LCD_M(HIGH);
    Nop();
    Nop();
    LATD = KS0108B_Data;
    Nop();
    Nop();
    E_LCD_M(LOW);
    Nop();
    E_LCD_M(HIGH);
    RW_LCD_M(HIGH);
    RS_LCD_M(HIGH);
}
```

di ogni ciclo macchina richiede 100nSec.

Ricordiamo che i PIC 18Fxxx hanno un PLL interno il quale può essere impostato per moltiplicare per 4 il valore di frequenza di oscillazione del quarzo esterno.

Nel nostro caso esso è da 10 MHz, quindi la frequenza di clock interna sale a 40 MHz. Come di consueto, la vera frequenza operativa dei PIC è ottenuta dividendo per 4 il valore del clock interno, ottenendo di conseguenza 10 MHz.

Se volesse avere ulteriori dettagli relativamente al PLL, potete consultare il data-sheet del PIC18F4620.

Vediamo ora come impostare l'indirizzo "Y Address". Come descritto in precedenza, di questo comando si possono usare solo i 6

bit meno significativi per un massimo di 64 locazioni di memoria. Quindi il comando assume una parte fissa a cui va sommato un valore variabile che identifica l'indirizzo di memoria in cui leggere o scrivere. La parte fissa viene identificata dalla costante "KS0108B_SET_ADD" e vale 0x40, mentre l'indirizzo vero e proprio viene identificato dalla variabile "KS0108B_Y_Address".

Quindi la routine è composta da una chiamata a funzione alla quale viene passato un codice ottenuto eseguendo un'operazione di "OR" tra la costante "KS0108B_SET_ADD" e la variabile "KS0108B_Y_Address" (vedi Listato 7).

Vediamo ora come impostare l'indirizzo "X Address" o indirizzo di pagina. Il concetto è identico a quanto descritto pri-

ma: la sola differenza sta nel codice da inviare. Anche in questo caso abbiamo una costante "KS0108B_SET_PAGE_X_ADD" e una variabile "KS0108B_Page" la quale può assumere i valori da 0 a 7 in quanto solo gli ultimi 3 bit sono utilizzabili.

La costante "KS0108B_SET_PAGE_X_ADD" ha valore 0xB8 (vedi Listato 8).

Vediamo ora come impostare l'indirizzo "Z Address" necessario per eseguire uno scroll del display rispetto all'asse Y. In questo caso abbiamo una costante "KS0108B_SET_START_LINE" e una variabile "KS0108B_Start_Line". La costante vale 0xC0 mentre la variabile può assumere

Listato 7

```
void Set_Address_KS0108B(void)
{
    Write_Comm_KS0108B(KS0108B_SET_ADD | KS0108B_Y_Address);
}
```

Codice comando da inviare al controller grafico.

valori che vanno da 0 a 63 in quanto possono essere impostati solo i 6 bit meno significativi.

Il codice da inviare successivamente al controller grafico viene ricavato eseguendo un "OR" tra la costante "KS0108B_SET_START_LINE" e la

Listato 8

```
void Set_Page_KS0108B(void)
{
    Write_Comm_KS0108B(KS0108B_SET_PAGE_X_ADD | KS0108B_Page);
}
```

Codice comando da inviare al controller grafico.

variabile "KS0108B_Start_Line", come riportato nel Listato 9.

Analizziamo ora la routine con cui viene inviato un dato al controller grafico. Essa è molto simile alla routine per l'invio di un comando: differisce solo nel modo in cui viene mosso il segnale di controllo RS (Listato 10).

A questo punto abbiamo definito le routine base per la gestione del controller grafico. Non ci resta che vedere le routine necessarie a visualizzare le immagini sul display stesso, iniziando con la descrizione di quella visibile nel Listato 11.

Listato 9

```
void Set_Start_Line_KS0108B(void)
{
    Write_Comm_KS0108B(KS0108B_SET_START_LINE | KS0108B_Start_Line);
}
```

Codice comando da inviare al controller grafico.

Listato 10

```

void Write_Data_KS0108B(uchar KS0108B_
Data)
{
    E_LCD_M(LOW);
    Nop();
    Nop();
    RW_LCD_M(LOW);
    RS_LCD_M(HIGH);
    Nop();
    E_LCD_M(HIGH);
    Nop();
    Nop();
    LATD = KS0108B_Data;
    Nop();
    Nop();
    E_LCD_M(LOW);
    Nop();
    E_LCD_M(HIGH);
    RW_LCD_M(HIGH);
    RS_LCD_M(LOW);
}

```

La funzione che si occupa dell'invio dei dati al controller KS0108B è identica alla funzione per inviare un comando. L'unica eccezione riguarda il segnale di controllo RS. Il quale deve essere a livello logico alto. Anche in questo caso la funzione è suddivisibile in due parti distinte.

1

2

Per prima cosa viene impostato l'indirizzo di partenza per la lettura dei dati dalla memoria EEPROM tramite bus I²C, nonché il controller (vedere l'istruzione *"if (Chipset != 1)"* all'inizio della funzione) che deve riceverli per memorizzarli nella pagina della memoria RAM.

Se il controller è il numero 1 si parte dall'indirizzo assegnato **"Address_EEPROM"**, mentre se il controller è il numero 2 si deve partire con uno spiazzamento di 64 byte rispetto all'indirizzo di partenza **"Address_EEPROM"**.

Quindi durante la lettura della EEPROM occorre "ragionare" a banchi di 64 byte ciascuno. Supponiamo di volere caricare la prima metà dell'immagine, e quindi di memorizzarla nella memoria RAM del controller 1. Si comincia a leggere la memoria EEPROM partendo dall'indirizzo **"Address_EEPROM"** e si continua fino ad

arrivare a un massimo di 64 byte di dati che vengono caricati nella prima pagina della memoria RAM del controller 1 (ricordiamo che ogni pagina di RAM occupa esattamente 64 Byte). Conclusa la pagina 1 è necessario passare alla pagina 2 della RAM sempre del primo controller. Quindi occorre programmarlo per puntare al primo byte di RAM della seconda pagina e incrementare il puntatore alla memoria EEPROM di 64 byte. Conclusa la scrittura della pagina 2 si passa alla pagina 3 e così via fino a leggere tutti i byte necessari a riempire le otto pagine di memoria RAM del controller 1. L'immagine di Figura 11 mette in evidenza quanto detto. I byte che compongono la seconda parte dell'immagine si trovano esattamente in mezzo ai banchi da 64 byte appena letti per il controller numero 1 (parte in violetto chia-

Listato 11

```

void Write_RAM_KS0108B(uchar Chipset, uint Address_EEPROM)
{
    uchar temp_count;
    KS0108B_Page = KS0108B_Page_0;

    if (Chipset != 1)
    {
        CS2_LCD_M(HIGH);
        I2C_EEPROM_Add.u1 = Address_EEPROM + (uint)64;
    }
    else
    {
        CS1_LCD_M(HIGH);
        I2C_EEPROM_Add.u1 = Address_EEPROM;
    }

    do
    {
        KS0108B_Y_Address = 63;
        Test_KS0108_Busy();
        Set_Address_KS0108B();

        Test_KS0108_Busy();
        Set_Page_KS0108B();

        temp_count = 0 ;

        do
        {
            I2C_Process();
            Test_KS0108_Busy();
            Write_Data_KS0108B(I2C_EEPROM_Data[0]);
            I2C_EEPROM_Add.u1 += 1;
        } while (++temp_count != 64);

        I2C_EEPROM_Add.u1 += 64;

    } while(++KS0108B_Page != 8);

    CS1_LCD_M(LOW);
    CS2_LCD_M(LOW);
}

```

Impostazione pagina di partenza per scrittura in RAM. La costante **"KS0108B_Page_0"** identifica la pagina 0.

Si ricorda che i controller a bordo del display 64128A sono due. Quindi è necessario indicare quale dei due controller usare, compito demandato alla variabile **"Chipset"**. Quindi in questo **"If"** si abilita il controller di interesse e si imposta l'indirizzo di partenza in EEPROM per la lettura dei dati da inviare alla memoria RAM del controller selezionato.

Impostazione indirizzo **"Y Address"**. Queste istruzioni vengono richiamate per ogni pagina della memoria RAM. Si noti che è stato realizzato un loop che scrive su tutte le otto le pagine della memoria RAM del controller abilitato.

Impostazione indirizzo **"X Address"** o indirizzo di pagina.

Lettura dati da memoria EEPROM su bus I²C e trasferimento dei dati letti alla memoria RAM del controller abilitato. Ad ogni lettura dalla EEPROM il rispettivo puntatore viene incrementato di una unità.

Loop di indirizzo. Massimo 64 byte.

Loop di pagina. Massimo otto pagine.

ro della memoria EEPROM, vedere Figura 11).

Oltre a una routine per scrivere nella memoria RAM del controller, è stata realizzata una routine che “pulisce” la memoria RAM del controller selezionato: in pratica scrive 0x00 in tutte le locazioni. Il concetto è quasi identico alla routine di cui abbiamo già parlato, senza però la necessità di andare a leggere la memoria EEPROM. Il codice è descritto nel Listato 12.

L'ultima routine si occupa di scrivere un font (Carattere o simbolo) in un punto qualsiasi della memoria RAM (Listato 13). Questa funzione richiede l'utilizzo di svariati parametri (in neretto); infatti è necessario indicare: l'indirizzo di partenza in EEPROM per leggere il font “**Address_EEPROM**”, il controller a cui inviare il font “**Chipset**”, il numero di pagina in cui scrivere “**Page**”, l'indirizzo di partenza da cui iniziare a scrivere in tale pagina “**Col_Address**”, la larghezza “**Width**” e l'altezza “**Height**” del font stesso. L'altezza del font è un multiplo di 8 pixel, mentre la larghezza può arrivare a un massimo di 64 pixel.

Gestione dei pulsanti

A questo punto abbiamo esaurito l'argomento relativo al codice di gestione del display. Manca solo una rapida descrizione delle funzioni svolte dai cinque pulsanti presenti sulla demoboard, visibili nella Figura 12. Come si può osservare, essi sono disposti a croce in modo da formare una specie di Joystick e, a seconda del display inserito (quindi più in generale a seconda del controller), svolgono funzioni diverse. Nel caso del display di cui stiamo parlando, le funzioni associate ai tasti sono le seguenti:

- **RIGHT**: richiama immagine successiva. Nella memoria EEPROM sono memorizzate fino ad un massimo di 4 immagini richiamabili utiliz-

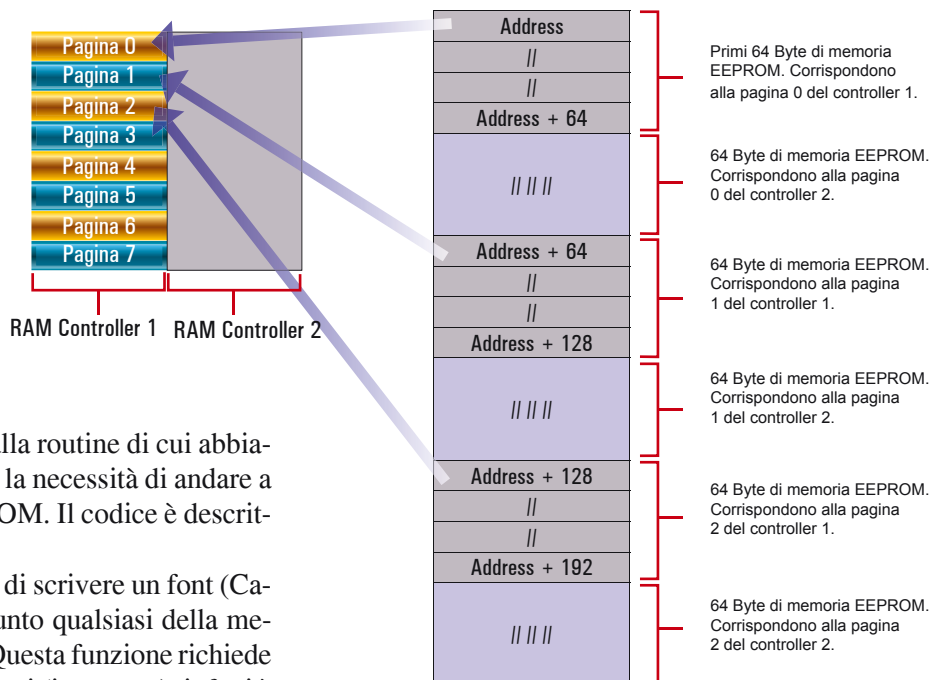


Fig. 11

zando il pulsante “RIGHT”.

Dopo la quarta immagine vengono mostrati i due font memorizzati nella memoria EEPROM (vedi Figura 13).

Listato 12

```
void Clear_RAM_KS0108B(uchar Chipset)
{
    uchar temp_count;

    KS0108B_Page = KS0108B_Page_0;
    KS0108B_Y_Address = 0;

    if (Chipset != 1)
    {
        CS2_LCD_M(HIGH);
    }
    else
    {
        CS1_LCD_M(HIGH);
    }

    do
    {
        Test_KS0108_Busy();
        Set_Address_KS0108B();

        Test_KS0108_Busy();
        Set_Page_KS0108B();

        temp_count = 64 ;

        do
        {
            Test_KS0108_Busy();
            Write_Data_KS0108B(0x00);
        } while (--temp_count != 0);

    } while(++KS0108B_Page != 8);

    CS1_LCD_M(LOW);
    CS2_LCD_M(LOW);
}
```

Selezione del corretto controller per azzerare il contenuto della memoria RAM.

Impostazione indirizzo “Y Address”. Queste istruzioni vengono richiamate per ogni pagina della memoria RAM. Si noti che è stato realizzato un loop che scrive su tutte le otto pagine della memoria RAM del controller abilitato.

Impostazione indirizzo “X Address” o indirizzo di pagina.

Loop di indirizzo. Massimo 64 byte.

Loop di pagina. Massimo otto pagine.

- **LEFT**: richiama immagine precedente. Come per il comando right, viene richiamata una delle quattro immagini contenute nella memoria EE- ➤

Listato 13

```

void Write_Font_RAM_KS0108B(uint Address_EEPROM, uchar Chipset,
uchar Page, uchar Col_Address, uchar Width, uchar Height)
{
    uchar temp_count;
    KS0108B_Page = Page;
    KS0108B_Y_Address = Col_Address;

    if (Chipset != 1)
    {
        CS2_LCD_M(HIGH);
    }
    else
    {
        CS1_LCD_M(HIGH);
    }

    I2C_EEPROM_Add.u1 = Address_EEPROM;
    temp_count = Width;

    do
    {
        Test_KS0108_Busy();
        Set_Address_KS0108B();

        Test_KS0108_Busy();
        Set_Page_KS0108B();

        do
        {
            I2C_Process();
            Test_KS0108_Busy();
            Write_Data_KS0108B(I2C_EEPROM_Data[0]);
            I2C_EEPROM_Add.u1 += 1;
        } while (--temp_count != 0);

        if (Height > 1)
        {
            temp_count = Width;
            KS0108B_Page = KS0108B_Page + 1;
            KS0108B_Y_Address = Col_Address;
        }

    } while(--Height != 0);

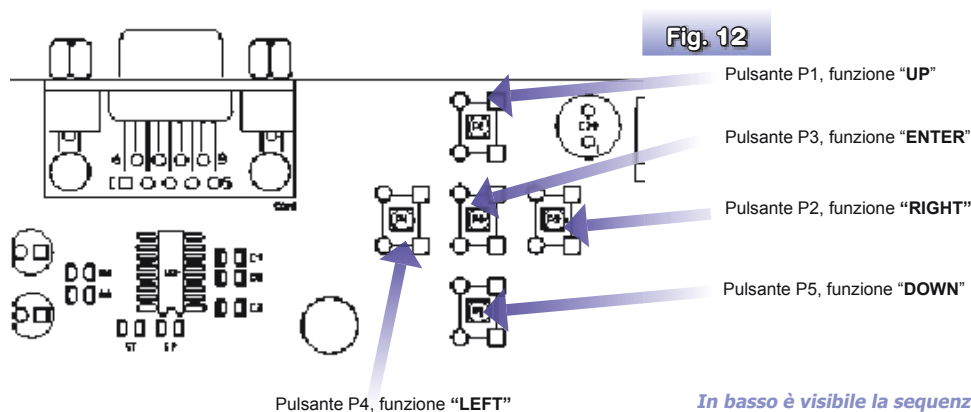
    CS1_LCD_M(LOW);
    CS2_LCD_M(LOW);
}

```

PROM. Dopo la prima immagine vengono mostrati anche i due font (vedi Figura 13);

- **UP**: richiama la funzione *Incrementa variabile “KS0108B_Start_Line”*, quindi esegue uno scroll verso l’alto dell’immagine;
- **DOWN** richiama la funzione *Decrementa variabile “KS0108B_Start_Line”*, quindi esegue uno scroll verso il basso dell’immagine;
- **ENTER**, con questo display, o più in generale con questo controller, non svolge alcuna funzione.

L’immagine in Figura 13 mette in evidenza il passaggio da un’immagine alla successiva utilizzando i pulsanti **“RIGHT”** e **“LEFT”**. A questo punto possiamo ritenere conclusa la spiegazione del display 64128A e del relativo controller KS0108B: i listati d’esempio dovrebbero consentire a chiunque di iniziare ad utilizzare correttamente questi dispositivi. Il prossimo mese ci occuperemo del display grafico 64128E e del controller KS0713.



A sinistra è riportato il dettaglio dei tasti per il controllo dei differenti tipi di display. È stata scelta la disposizione a croce, simile ad un joystick, perché in questo modo l’identificazione delle funzioni UP, DOWN, LEFT, RIGHT e ENTER è facilmente intuibile.

In basso è visibile la sequenza in cui le varie immagini vengono alternate sul display, in seguito alla pressione dei tasti LEFT e RIGHT.

