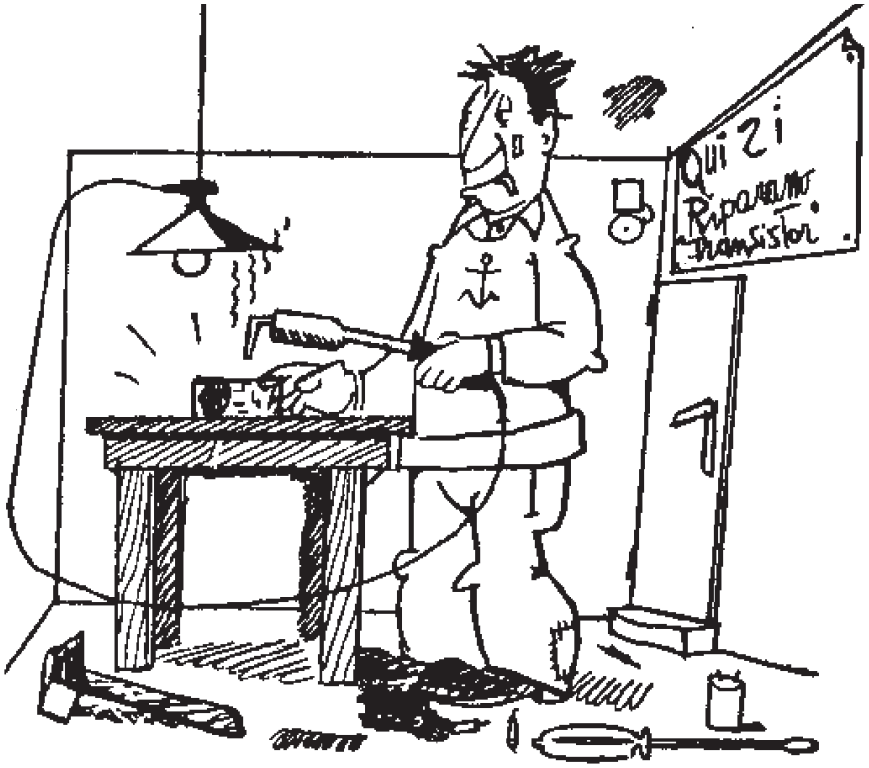


Questo articolo è stato pubblicato su....

ELETTRONICA FLASH

RESET PER BBS OVVERO FBBKIT

Daniele Cappa

Un insieme di componenti hardware per rendere più versatile e più sicuro il programma di F6FBB, e a seguire un sistema di reset hardware del tutto autonomo e assolutamente inedito, più una versione di MUX tratto da info dello stesso F6FBB.

Reset hardware

È un oggetto ormai supercollaudato, il prototipo funziona da dieci anni (non è un errore) senza nessun problema, malgrado la realizzazione decisamente spartana (Foto 1).

Chi di noi usa un PC in casa sa bene come sia possibile che il programma in uso si "pianti" quando noi non siamo presenti. Il caso più evidente è nell'uso di BBS amatoriali, o telefonici, la cui sorverglianza ha inevitabilmente dei buchi di alcune ore. Il sistema più veloce per far ripartire la macchina, anche se non risolve gli eventuali problemi del programma, è quello di spegnerla e riaccenderla. Inoltre ci si propone un sistema che sia indipendente dal programma in uso attualmente, che può essere rimpiazzato, e dal tipo di computer; deve essere un sistema hardware esterno al PC, che sia installabile senza problemi da chiunque e che non comporti neppure una saldatura sulla macchina in uso.

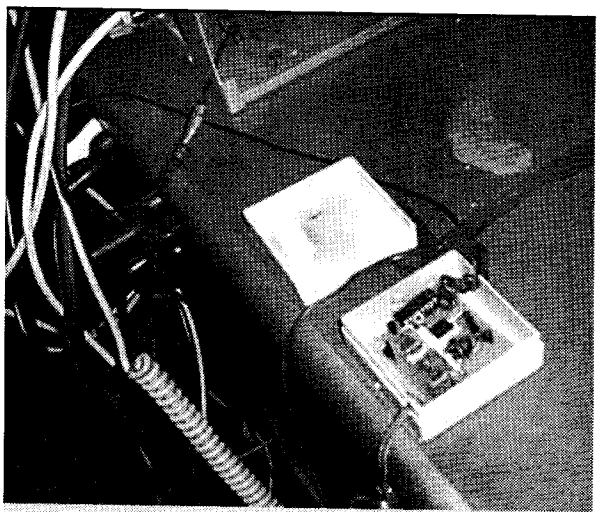


Foto 1 - Reset aperto: inserito in un contenitore di recupero, la confezione di un cuscinetto a sfere, accanto alla parete del PC si vede il gruppo LED-fotodiodo racchiuso in un pezzetto di guaina termorestringente.

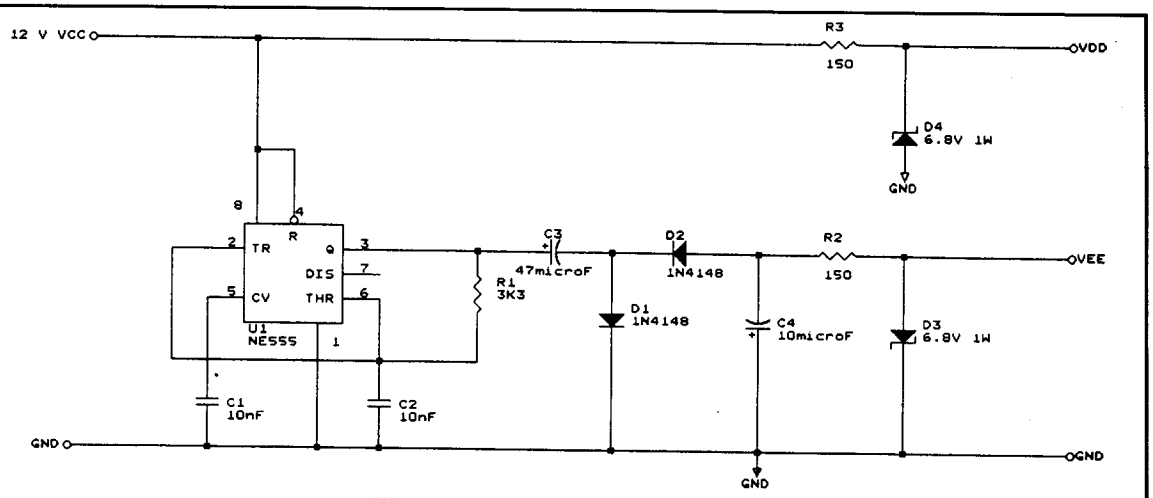


figura 1 - Alimentatore per MUX.

R1 = 3,3kΩ
R2 = R3 = 150Ω
C1 = C2 = 10nF multistrato
C3 = 47µF/16V elettr.
C4 = 10µF/16V elettr.
D1 = D2 = 1N4148
D3 = D4 = 6.8V/1W zener
U1 = NE555

I BBS amatoriali, inseriti in rete, effettuano regolari operazioni di forward che impediscono alla macchina di restare inattiva per più di pochi minuti; questa situazione ci permette di sfruttare il LED di accesso dell'hard disk per il nostro scopo. In pratica inseriamo un interruttore automatico sulla linea a 220V che alimenta il computer; quest'ultimo viene "spento" se il disco rigido resta inattivo per più di alcune decine di minuti (30-40 minuti, o più se necessario), per venire poi riacceso circa un minuto dopo. L'uso si estende a molte altre applicazioni, sia prelevando il segnale di attività dal floppy, sia non prelevandolo del tutto, se la nostra intenzione è di spegnere il PC dopo un tempo predeterminato. Le temporizzazioni sono ampiamente modificabili, sia dal punto di prelievo delle uscite, sia dal tempo di clock del contatore.

Il circuito si compone di un generatore di clock, il suo periodo è regolabile da 1 a 1.3 secondi (nulla impedisce di aumentarlo a piacere per ottenere tempi finali molto lunghi) e da un contatore binario a 12 bit (CD4040).

Il contatore viene resettato (tramite un transistor e un fotodiodo) nel momento in cui si accende il LED di lettura dell'hard disk; in questo modo abbiamo evitato ogni intervento hardware sul computer. Il collegamento si limita ad inserire tra il PC e la rete i contatti del relè del timer e, con due centimetri di guaina termorestringente, il fotodiodo "davanti" al LED di lettura dell'hard disk. Per avere i due tempi di commutazione, quello PC acceso relativamente lungo e quello "PC spento" corto ma non troppo, abbiamo utilizzato una porta logica AND realizzata con 2 diodi al germanio (NON al silicio).

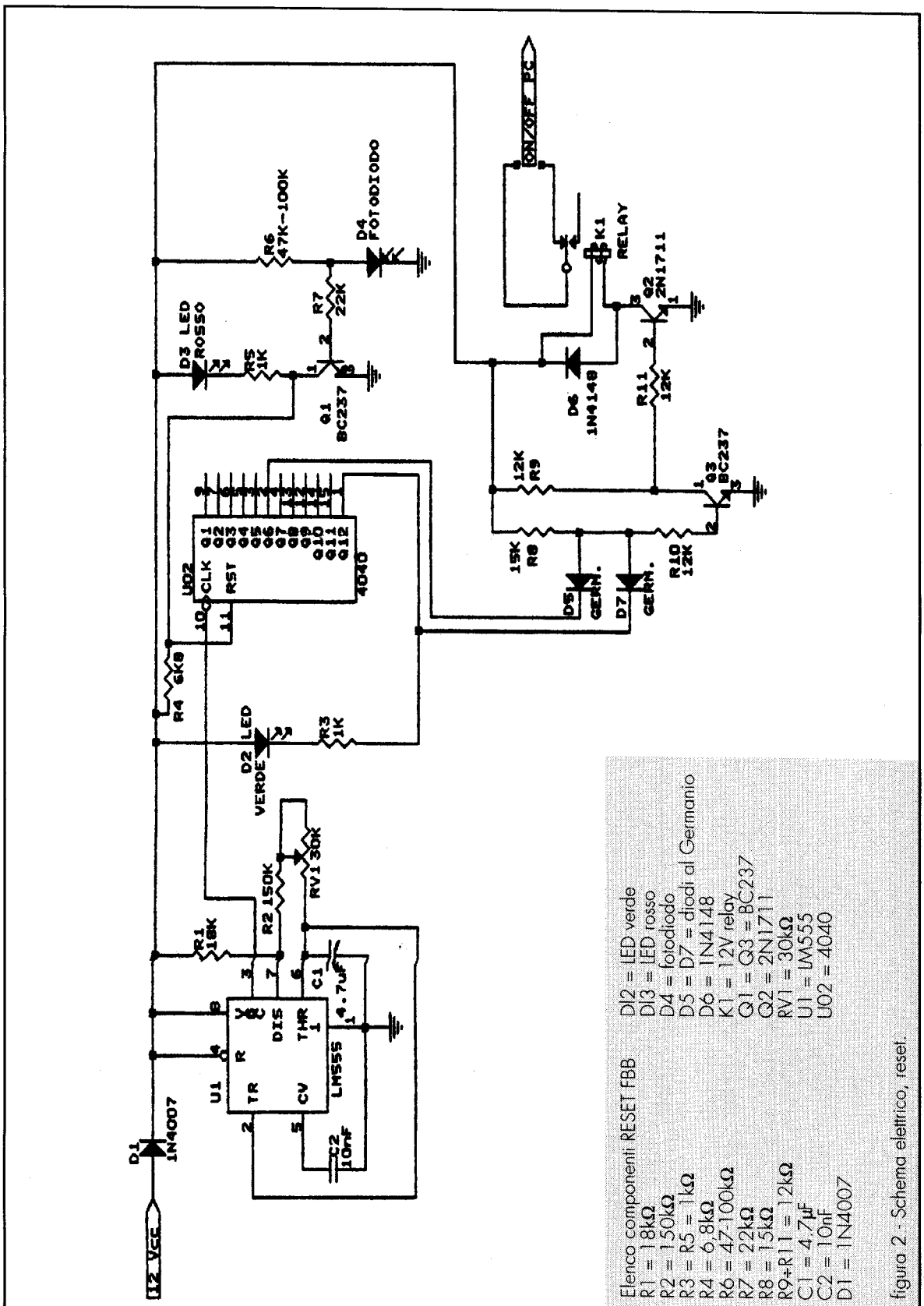
Nulla vieta di impiegare un CD4081 per ottenere la stessa funzione.

Vediamo il funzionamento:

Si è usato un NE555 in configurazione astabile quale generatore di clock, cui fa seguito un CD4040 che fornisce alle sue uscite il numero binario degli impulsi trascorsi dall'ultimo reset. Il reset del contatore avviene quando riceve un impulso a livello logico 1 sul pin di reset.

Appena acceso il timer il CD4040 inizia il suo conteggio e il relè è eccitato, affinché il relè torni in posizione di riposo è necessario che i due diodi al germanio abbiano ENTRAMBI il catodo a livello logico 1, dopo 32 impulsi di clock il bit 5 passa a 1, dopo altri 32 a zero e così via fino a che siano trascorsi 2048 + 32 impulsi, quando anche il bit 11 passa a livello logico 1; con questa condizione il transistor (BC237) va in saturazione, fino ad ora era sempre stato in interdizione, il 2N1711, non avendo più corrente di base, si presenta come un interruttore aperto e il relè si diseccita... Il PC è ora spento.

Questa condizione permane per altri 32 impulsi di clock: appena il bit 5 del 4040 torna a livello 0, i due transistor rieccitano il relè, Se queste condizioni non dovessero verificarsi ènecessario



Elenco componenti RESET FBB

R1 = 18kΩ
R2 = 150kΩ
R3 = R5 = 1kΩ
R4 = 6.8kΩ
R6 = 47-100kΩ
R7 = 22kΩ
R8 = 15kΩ
R9+R11 = 12kΩ
C1 = 4.7µF
C2 = 10nF
D1 = 1N4007
DI2 = LED verde
DI3 = LED rosso
D4 = fotodiodo
D5 = D7 = diodi al Germanio
D6 = 1N4148
K1 = 12V relay
Q1 = Q3 = BC237
Q2 = 2N1711
RV1 = 30kΩ
U1 = LM555
U02 = 4040

figura 2 - Schema elettrico, reset

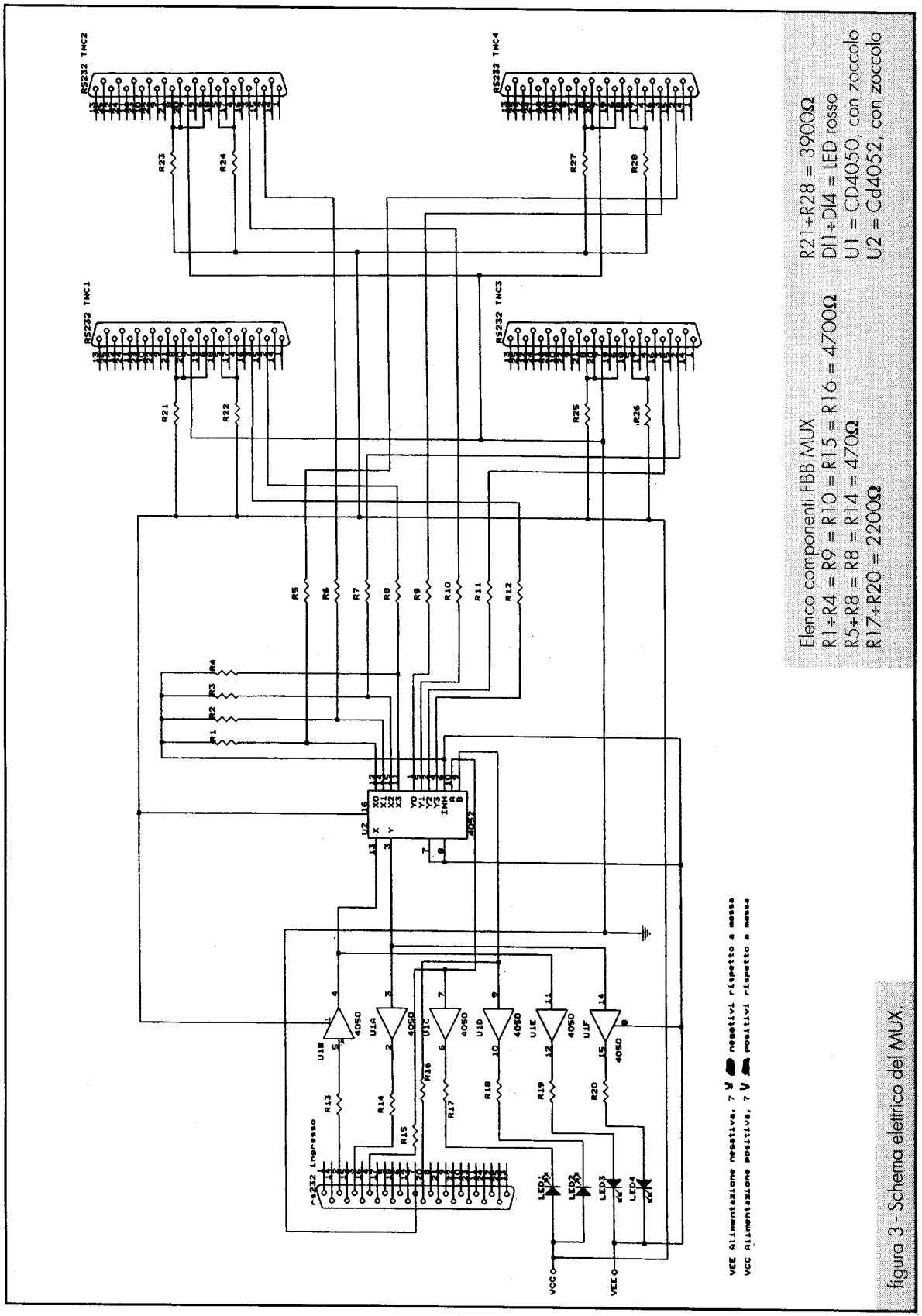


figura 3 - Schema elettrico del MUX.

aumentare i tempi, aumentando il periodo di clock (sostituendo il condensatore collegato al pin 6 del NE555 con uno di capacità maggiore) oppure, se deve essere aumentato solo il periodo OFF-ON del PC, e quindi quello in cui il LED dell'hard DEVE accendersi, basta spostare il catodo del diodo al germanio dal bit 5 (pin 2) al bit 6 (pin 4 del 4040) in questo modo la commutazione OFF-ON avviene ogni 64 impulsi di clock. Con i valori indicati si ottengono tempi di commutazione variabili (con il trimmer..) di 34-41S per il periodo OFF e di. 36-44 minuti per il periodo ON. Un prototipo molto artigianale funziona da anni su *i1ylm BBS*, prima su XT, poi su 286, quindi su due diversi 386 fino all'attuale pentium; ci ha risparmiato un buon numero di chilometri per soccorrere il BBS puntualmente inchiodato nel momento meno opportuno! Controllate bene il log del vostro BBS, al fine di verificare che il timer non provochi lo spegnimento della macchina quando non è necessario.

Il sistema di spegnere del tutto la macchina non è sicuramente tra i più tecnicamente evoluti, ma si è sempre dimostrato efficace; per chi usa il soft di F6FBB con TNC2 e firmware host mode è possibile l'uso di due relè, in cui il secondo priva dell'alimentazione i TNC che, senza batteria di backup, si resetteranno anche loro evitando problemi di partenza al soft del BBS.

Per il controllo del contatore abbiamo inserito 2 LED che permettono di controllare il buon funzionamento del CD4040, uno si attiva ad ogni impulso di reset, ripetendo così il LED dell'hard disk "capovolto". Questo LED è acceso quando il led dell'hard è spento e viceversa, l'altro ogni 4 impulsi di clock (bit 2), osservandoli si capisce se il sistema è operativo. La resistenza R6, da 47kΩ a 100kΩ, andrà scelta secondo il fotodiodo usato, controlleremo che la commutazione avvenga correttamente osservando D3, il LED rosso, che deve essere acceso quando il LED dell'hard è spento e viceversa.

Multiplex per FBB BBS

Il magnifico programma, di cui F6FBB è l'autore ha limiti di hardware, in modo particolare sulle seriali che il vecchio DOS è in grado di ospitare. Dalle due normali si sale relativamente volentieri a quattro, anche se si perviene a tale risultato solo modificando i due IRQ e sfruttando quelli delle porte parallele che abitualmente sono poco o nulla usate su un BBS.

Questo fatto limita a quattro il numero di TNC, e quindi di porte, che il sistema è in grado di gestire. Il problema è stato risolto dallo stesso autore con una scheda di multiplex che offre la possibilità di collegare fino a quattro TNC ogni porta seriale (Foto 2).

I due esemplari costruiti sono stati entrambi realizzati con cablaggi a filo, il primo sfruttando un contenitore plastico e realizzando la necessaria alimentazione negativa con il solito 555; il secondo sfruttando la parte di alimentazione e il contenitore di un non ben definito oggetto (Foto 3) sulla cui parte posteriore spiccavano quattro connettori canon a 25 poli adatti al nostro uso, e risparmiando così non poco lavoro per realizzare la foratura del pannello posteriore.

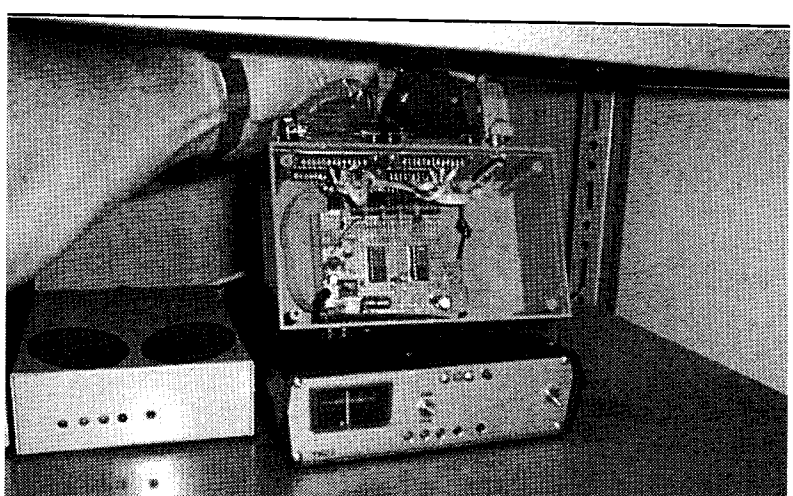


Foto 2 - Mux ad alimentazione singola, con il classico NE555, assieme ai TNC2 che controlla, un 1200 baud AFSK (con gli occhi) mentre l'altro è dotato di modem PSK, sempre a 1200 baud.

Il primo realizzato è stato quello con il NE555 che è stato messo insieme nel '91 ed è in uso "solamente" dai primi mesi del '95, mentre il secondo è stato assemblato ed è in uso dai primi giorni di giugno '98.

L'installazione del MUX è molto semplice, si tratta di modificare la riga di lancio dei driver di seriale sostituendo il vecchio "ESS" con il nuovo "ESSMUX",

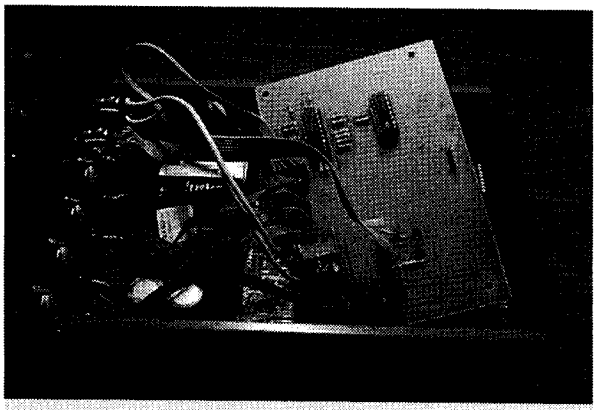


Foto 3 - MUX inserito nel contenitore di cui è stato recuperato l'alimentatore, e in evidenza collegamenti al pannello posteriore.

compreso nel pacchetto del software di F6FBB, e la modifica di alcune righe nel file PORT.SYS, pubblicato di seguito.

I TNC in uso sul MUX dovranno essere TNC2 equipaggiati con firmware host mode stile DED statunitense o uno dei vari TFxx Nord<>Link tedeschi. La velocità via seriale non dovrà essere troppo elevata e si farà in modo da sfruttare il MUX sulle porte a velocità più bassa. Indicativamente quattro porte a 1200 baud lavorano molto bene sotto MUX anche se due di queste sopportano un traffico decisamente sostenuto. Nell'esempio la velocità via seriale è stata tenuta a 9600 baud, valore di tutto rispetto usando quattro TNC che lavorano sul canale radio a 1200 baud.

L'esemplare più giovane sta ora lavorando con tre porte, due a 1200 baud e una a 4800 baud machester verso la radio. Personalmente ritengo che le porte a velocità più alta, 9600 baud o più, sia bene usarle singolarmente anche se il MUX dovrebbe reggere bene il traffico decisamente più intenso a cui è sottoposta una porta operante a alta velocità verso il canale radio.

Il driver usato, ESSMUX, si incarica di inviare ai pin 4 e 20 della seriale del PC i comandi di commutazione del mux. Questi, dopo essere stati bufferizzati dal 4050, giungono ai due pin di indirizzo del 4052 che è un multiplex a due vie per quattro canali che "collega" i pin dati della seriale (due e tre) con i corrispondenti pin del connettore di uscita selezionato dai pin 4 e 20.

Attenzione all'alimentazione del tutto che deve essere duale, da 6 a 7V positivi su VCC e da 6 a 7V negativi rispetto a massa sul pin contrassegnato con VEE. È bene NON eccedere oltre i 7V di alimentazione per ramo, pena il superamento dei 15V di alimentazione massima consentita per i componenti in tecnologia CMOS.

Per chi non dovesse disporre della alimentazione duale è possibile realizzare il solito oscillatore con NE555 che genera la tensione negativa necessaria al funzionamento del tutto, anche questo realizzato con tecnica filata e visibile nel prototipo senza il trasformatore (Foto 4).

Questo semplice circuito permette di alimentare il MUX con un comune alimentatore in grado di fornire da 9 a 12Vcc, anche utilizzando un modello a spina.

Sulla piastra del MUX che usa il contenitore e l'alimentatore di recupero sono visibili alcuni componenti, tra cui un relè, di cui non si è fatto alcun cenno. Sono le interfaccie di alcuni programmi realizzati da colleghi OM e che sono state assemblate insieme al MUX.

Bibliografia

- Documentazione originale del programma FBB BBS, di F6FBB.
- Data sheet Fairchild, per i chip utilizzati nelle due realizzazioni.
- Data sheet National per il "solito" NE555.

Ringraziamenti

- IK1MJJ Aldo Carlo, sta usando la versione di MUX più recente.
- I1YLM Bruno, la cui stazione è illustrata e presso

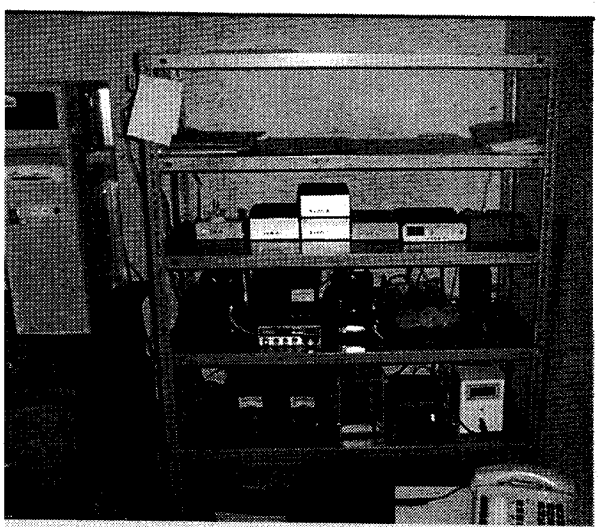


Foto 4 - Veduta d'insieme del sistema, il MUX è a destra sullo scaffale che ospita tutti i TNC.

```
# FBB7.00
# File for programming of channels and TNCs.
#
# Ports : How many ports (COM1, COM2, Etc...)
# TNCs : How many TNCs and modems in use. With multiplexer
# there can be up to 4 TNCs per port.
#
#Ports TNCs --  esempio di una seriale con quattro tnc sotto mux
1      4
#
#In WinFBB ONLY THESE interfaces are available:
# Interface : 2 = BPQ-node (BPQ in AA4RE-mode)
#             4 = DRSI
#             5 = TFPCR/TFPCX interface.  Interrupt MUST be 0xFD or the same
#                 as stated in INIT.SRV, if any..
#             6 = Windows-driver, replaces both ESS, ESSKAM and FBBIOS.
#             7 = TCP/IP.  Needs WINSOCK.DLL.  Put port-address as 17.
#                 TNC-emulation is T (see below)
#             8 = TFWin.dll (Only WinFBB32)
# BEWARE: The old interface 1 and 3 will NOT be used in WinFBB.  Interface 6
#         replaces both.  (FBBCOMM.DRV).  Neither ESS nor FBBIOS can be
#         used with WinFBB !
#
#In LinFBB ONLY this interface is available:
# Interface   9 = Linux.  Can work via serial port (D),via AX25 domain
#                 socket (X) or via Telnet port (T).
#
#In DosFBB ONLY THESE interfaces are available:
# Interface : 1 = Use external COMBIOS-driver (MBBIOS, ESS etc)
#             2 = BPQ-node v 4.05 and up (BPQ in AA4RE-mode)
#             3 = Telephone-modem with FBBIOS
#             4 = DRSI card with driver
#             5 = TFPCR/TFPCX interface.  Interrupt MUST be 0xFD
#
# Address   : Address of port in hexadecimal (Needed for multiplexer).
#             In LinFBB:
#             Address is the device name (/dev/cua0).
#             Be sure you have the rights to access to the device (rw-rw-rw-).
#             When using kernel AF_AX25 socket, address is not used.
#             When using Telnet, address is the Telnet port in Hex (Hex 17=Tel
net port 23)
# Baud      : Ports baud rate. Ignored by BPQ, kernel AF_AX25 socket and Telnet.
#
# Use same number of lines as number of ports.
#
#Com Interface Address (device)   Baud
1    1         3F8                9600
#2    1          2F8                9600
#
# TNC     : Number on TNC in use.  Use 0 for file-forward !
# NbCh    : Number of channels I want to use in the TNC.
#           Maximum available channels depend on firmware.
# Com     : Number of the COM-port. Com1, Com2 etc.
# MultCh  : Number of channel if port-multiplexer is used, otherwise 1.
#           In DRSI use values from 0 to 7, by KAM use 1/VHF and 2/HF.
#           With BPQ first TNC must have MultCh 0, the next 1, etc.
#           When using kernel AF_AX25 socket in Linux, MultCh is the
#           interface name (eg: ax0)
# Paclen  : PACLEN on this TNC.
# Maxframe: The maximum nb of frames the TNC will send at a time.
# NbFwd   : Number of channels for OUTGOING forward at same time
# MxBloc  : Size of forward-block in kb.
# M/P-Fwd : Minute of the hour for start of forward, and period
#           (how many minutes between each forward-start).
```

```
# Port mode, one of these:
#           B : BBS-mode.
#           G : "Guest"-mode.
#           U : Normal-mode.
# Type host-mode, one of these:
#           D : WA8DED
#           K : KAM hostmode.  Must use ESSKAM driver.
#           P : PK-232
#           Q : BPQ v 4.x
#           T : Ethernet/TCP-IP
#           X : AX25 domain socket (for Linux)
# Addition: One or more of these letters can be used too:
#           L : Send unproto beacon after each arriving mail.
#           M : Telephone-modem.
#           Y : Yapp allowed on this QRG.
#           W : Gateway allowed TO this QRG.
#           R : Modem port allowed in Read-only mode.
# Freq.   : Text to describe this port (max 9 characters, no space)
#
# Same number of lines as TNCs:
#
#TNC NbCh Com MultCh Pacln Maxfr NbFwd MxBloc M/P-Fwd Mode Freq
1    4    1   1      250   7     1     10     30/60   UDYW 144.xxx
# primo tnc,   com 1, porta 1 del mux
2    4    1   2      250   7     1     10     30/60   UDYW 433.xxx
# secondo tnc, com 1, porta 2 del mux
3    4    1   3      250   7     1     10     30/60   UDYW 435.xxx
# terzo tnc,   com 1, porta 3 del mux
4    4    1   4      250   7     1     10     30/60   UDYW 144.xxx
# quarto tnc,  com 1, porta 4 del mux
#
# Su tutte le porte sono state abilitati 4 canali contemporanei (NbCh)
#
# End of file.
#
```

cui funziona il reset da 10 anni e il MUX da poco più di tre.

- Tutti coloro che con me condividono la fatica di gestire un bbs: iw1biy marco e tutti i sysop.
- IW1BNV Roberto, a cui devo l'uso della fotocamera digitale.