UNIVERSITÀ DEGLI STUDI DI FIRENZE
FACOLTÀ DI INGEGNERIA

DIPLOMA UNIVERSITARIO IN INGEGNERIA ELETTRONICA

Anno Accademico 1996/1997

RELAZIONE SU ATTIVITÀ DI TIROCINIO
SVOLTO PRESSO L.A.M. Technologies SUL TEMA:

"Progettazione hardware e software di un terminale industriale standard VT100 - VT52 con display LCD"

Candidato: Claudio Gambelli

Relatori:

Chiar.mo Prof. Patrizio Piazzesi

Sig. Riccardo Landi

INDICE

1. INTRODUZIONE

Oggetto della presente tesi è la pregettazione hardware e software di un terminale industriale in grado di emulare gli standard VT100 e VT52.

La visualizzazione dei caratteri ricevuti avviene mediante un display LCD retroilluminato mentre la tastiera è realizzata in configurazione QWERTY.

Il terminale realizzato trova numerose applicazioni nel settore industriale quale terminale utente per la gestione di macchine automatiche, per la visualizzazione di messaggi di allarme o come terminale locale di programmazione. In futuro non si esclude tuttavia l'utilizzo del terminale in settori diversi quali il controllo accessi, ecc.

Durante tutte le fasi della progettazione sono state tenute in prima coniderazione esigenze di costo, affidabilità e reperibilità dei componenti impiegati. Per minimizzare il numero di parti presenti sulla scheda si è fatto uso delle moderne logiche programmabili PAL (Programmable Array Logic).

Lo sviluppo del software è avvenuto pensando a un pacchetto aperto, in grado di adattarsi facilmente a un diverso tipo di display o ad una diversa tastiera con modifiche minime.

A seguito dei calcoli teorici è sempre seguita una verifica su campo che mi ha dato la possibilità di affinare la pretica di utilizzo della strumentazione di laboratorio. Durante la realizzazione del progetto ho avuto modo, in particolare, di usare un oscilloscopio digitale e una scheda di emulazione collegata serialmente ad un PC.

Per lo sviluppo del software, interamente in assembler, è stato usato un compilatore della IAR oltre a vari programmi per velocizzare le fasi di compilazione e test del programma.

Gli schemi elettrici sono stati realizzati mediante il CAD elettronico OrCAD mentre per la programmazione dei componenti PAL è stato usato il programma PALASM.

2. GENERALITÀ SUL PROGETTO

Il terminale è composto principalmente da un display grafico LCD e da una tastiera per l'immissione dei dati. In questa versione del progetto il terminale dovrà essere in grado di emulare gli standard VT52 e VT100, per cui è risultato sufficiente farlo lavorare nella sola modalità TESTO, ossia sarà permessa la sola visualizzazione dei caratteri appartenenti alla tabella ASCII. In futuro sarà possibile con sole modifiche del software poter far lavorare il terminale in modalità grafica potendo visualizzare grafici, icone e testi di varie altezze, estendendone così l'utilizzo ad applicazioni particolari.
Il terminale da realizzare deve possedere così le seguenti caratteristiche:

- Display grafico Hitachi LMG7400 retro-illuminato risoluzione 128 x 240 pixel
- Regolazione del contrasto del display, sia via software che tramite potenziometro esterno
- Compensazione termica del contrasto
- Tastiera configurazione QWERTY
- Interfaccia seriale RS232
- Unica alimentazione di 24 Vdc +20% - 10%

Da un punto di vista di schema a blocchi, come si osserva da figura 2-1, il sistema che realizza il terminale è costituito da una unità centrale a microcontrollore, la quale ha come interfacce la tastiera, il Display LCD, e la seriale RS232, come ingressi riceve il sensore per la temperatura NTC, il potenziometro per la regolazione manuale del contrasto, possiede inoltre un'uscita di tipo PWM per realizzare la tensione per la regolazione del contrasto del display. Per quanto riguarda la memoria sia la EPROM per il software, che la RAM per i dati sono state realizzate esternamente al MCU per necessità di spazio.
Il sistema è alimentato con un alimentatore switching; il blocco PWM ha la funzione di convertire in analogico, ossia di integrare, il segnale PWM generato dal micro.

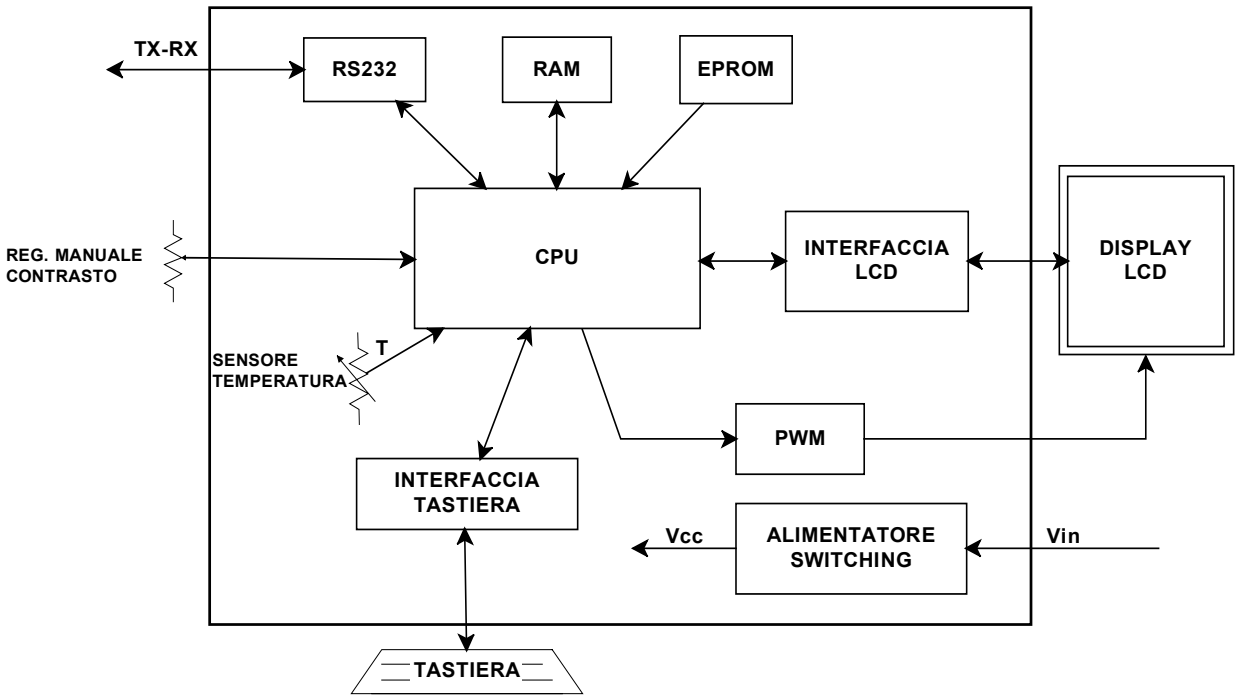


Figura 2-A

CPU:

La necessità di un interfaccia seriale e la presenza di un grosso numero di periferiche esterne, ha portato alla scelta del microcontrollore 68HC11 della Motorola, il quale avendo la possibilità di fornire esternamente BUS dati a 8 bit e BUS indirizzi ,capace di indirizzare 64Kbyte, ha permesso con la tecnica Memory mapped I/O di implementare l'interfaccia LCD, la tastiera e chiaramente RAM ed EPROM tutti sullo stesso BUS.

Il 68HC11 contiene al suo interno un Timer utilizzato in questo caso per generare il segnale PWM. Possiede un convertitore A/D utilizzato per acquisire i segnali analogici del potenziometro e del sensore di temperatura; rimangono inoltre disponibili ingressi e uscite digitali e l'interfaccia seriale SPI per futuri ampliamenti del sistema, il 68HC11 è così risultato il controllore più appropriato.

DISPLAY LCD:

Il display HITACHI LMG7400 utilizzato in questa applicazione è un 240 dot x 128 dot, permette di lavorare sia in modalità testo che modalità grafica, nel primo caso è sufficiente inviare il codice ASCII di un carattere per poterlo visualizzare, sfruttando il set di caratteri interno, altrimenti in modalità grafica occorre lavorare pixel per pixel.

Dato che il terminale deve visualizzare solo caratteri ASCII si è fatto lavorare il display in modo testo, lo schermo LCD è in grado di visualizzare in questa modalità 16 righe da 30 caratteri.

L'interfaccia LCD è risultata necessaria per accoppiare il microcontrollore al display, infatti benché esso possa essere interfacciato direttamente sul Bus tramite un proprio controller integrato, il MCU è risultato troppo veloce rispetto al display per cui è stata necessaria un'operazione di mantenimento dei dati tramite questa interfaccia.
Per quanto riguarda la regolazione del contrasto il display necessita anche di una compensazione termica per cui il segnale PWM che dovrà generare l'MCU dovrà dipendere oltre che dal potenziometro anche dal sensore di temperatura realizzato tramite un termoresistore NTC.
Sul display è anche inglobata una lampada CFL per l'illuminazione ed è compreso anche l'elevatore di tensione che però necessita di apposita alimentazione da 12 V.

TASTIERA:

La tastiera utilizzata dal terminale è del tipo QWERTY comprendente oltre ai tasti standard anche tasti funzione, è necessario inoltre che vengano lette anche pressioni contemporanee di 2 o più tasti.
La tastiera è costituita da una matrice di tasti ognuno dei quali premuto effettua un corto circuito fra la riga e la colonna associate ad esso, la lettura viene effettuata attivando una riga alla volta e leggendo la colonna corrispondente, l'interfaccia tastiera assolve così la funzione di utilizzare il Bus dati una volta per la decodifica delle righe e successivamente per la lettura delle colonne.

PWM:

Il blocco PWM è in realtà un semplice integratore che ha la funzione di fare la media del segnale PWM generato dal MCU, infatti variando via software il Duty-cicle si ottiene all'uscita di tale integratore una tensione continua variabile nel range necessario al display per la regolazione del contrasto.

RS232:

Il 68HC11 contiene al suo interno la SCI (Seriale Communication Interface), ossia un ricevitore e trasmettitore seriale universale, via software è possibile settare i vari parametri, fra cui il Baud rate, il n° di bit della trama e gli interventi sui vari tipi di errore. Per cui il formato dei dati seriale viene gestito interamente dal MCU, a livello fisico però per rispettare la standard RS232 occorre generare una tensione compresa fra +5 e +15 V per lo stato 0, e una tensione fra -5 e -15 V per lo stato 1, per realizzare tali tensioni si è dovuto così utilizzare l'interfaccia RS232 che non è altro che un integrato apposito accoppiato con alcuni condensatori in grado così di fornire tali livelli.

3. PROGETTAZIONE HARDWARE

3.1 Interfacciamento del microcontrollore al BUS dati e BUS indirizzi

Il microcontrollore 68HC11 per fornire il bus dati e bus indirizzi esternamente lavora in modalità Extended Multiplexed ossia le linee di indirizzo alte A8-15 vengono fornite separatamente mentre quelle basse A0-A7 vengono multiplexate con i dati, è necessario per cui congelare gli indirizzi mentre il bus dell'MCU è utilizzato per i dati, per fare questo si utilizza un Trasparent-Latch
74HC373 abilitato direttamente dall'MCU con il segnale AS (Address Strobe), come da figura.

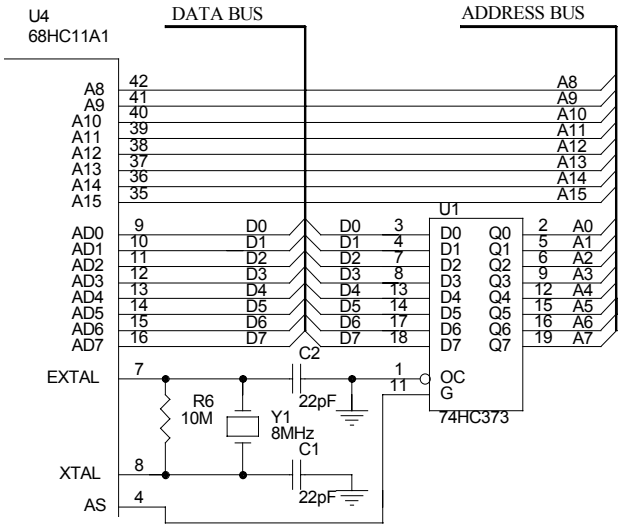


Figura 3-1

3.2 Definizione dello spazio di memoria

Occorre adesso definire lo spazio di memoria con cui lavora il microcontrollore 68HC11, il quale è in grado di indirizzare 64 Kbyte, in questo spazio devono risiedere la RAM, la EPROM, e le porte mappate in memoria, ossia l'interfaccia LCD e l'interfaccia tastiera. L'accesso all'interfaccia del display avviene in due modi o per scrivere istruzioni o per leggere o scrivere dati, per cui occorre dedicare due Byte uno per l'INSTRUCTION register e l'altro per il DATA register lo stesso per l'interfaccia tastiera per la quale occorre riservare un byte per il Latch-decoder di scrittura delle righe ROW keyboard ed un altro byte per il Buffer tree-state di lettura delle colonne COL keyboard. Per le porte mappate in memoria è stato lasciato libero 1Kbyte anziché soli 4 Byte per ridurre il numero di linee di indirizzo necessarie alla decodifica, questo

spazio è stato tolto nella parte dedicata alla EPROM riducendo così il suo spazio a 31Kbyte più che sufficiente per contenere il programma, questo spazio superiore al necessario inoltre può permettere in futuro di aumentare il numero di porte mappate in memoria. Lo spazio memoria sarà mappato come in figura 3-2.

Le linee di indirizzo necessarie per selezionare lo spazio I/O sono le A15-A10 per individuare il KByte e poi le A1 ed A0 per selezionare le quattro possibile porte; poiché non si sono utilizzate tutte le linee di indirizzo le quattro porte si troveranno ripetute in tutto l'intervallo del Kbyte ma questo non crea nessun problema dato che considerando tale spazio riservato non verranno mai generati gli indirizzi consecutivi.

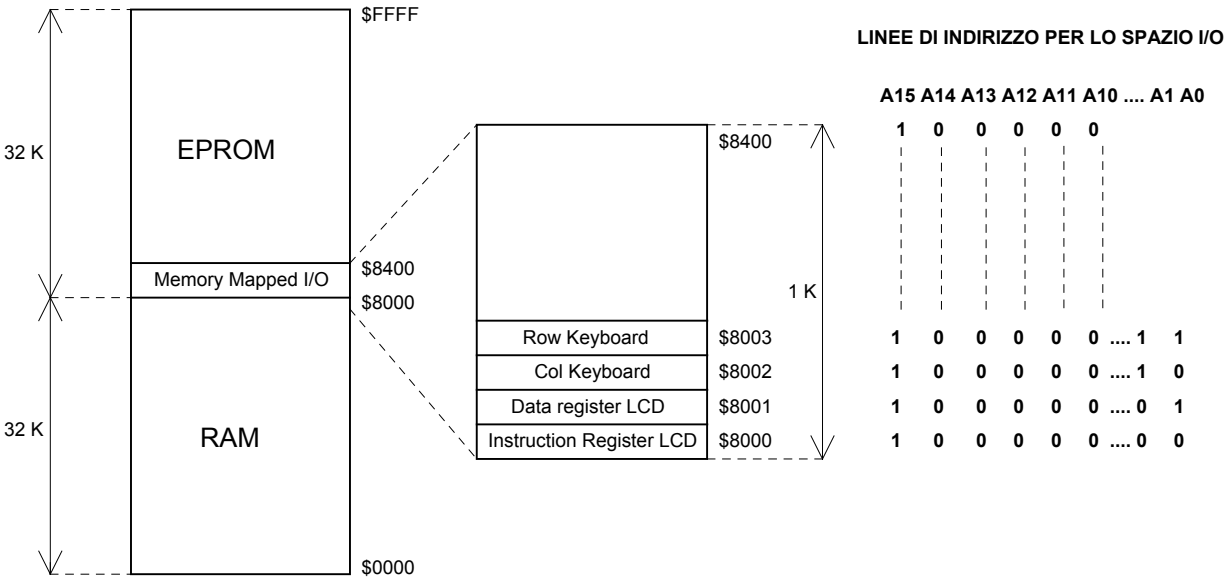


Figura 3-B

Per realizzare la logica combinatoria necessaria a decodificare gli indirizzi ed attivare così la RAM l'EPROM o le porte I/O la scelta migliore è stato utilizzare una logica programmabile ossia una PAL riducendo così lo spazio su stampato ma soprattutto riducendo il tempo di propagazione, dato che questo rimane invariato qualunque sia la complessità delle equazioni logiche, la PALCE16V8 della AMD è risultata sufficiente offrendo 10 ingressi e 8 uscite con un tempo di propagazione ingressi-uscite pari a 25 ns.

Oltre alle linee di indirizzo sopra citate la PAL dovrà avere in ingresso il Clock E e il segnale R/W generati dal MCU per sincronizzare così i segnali di abilitazione al ciclo di Bus del microcontrollore.

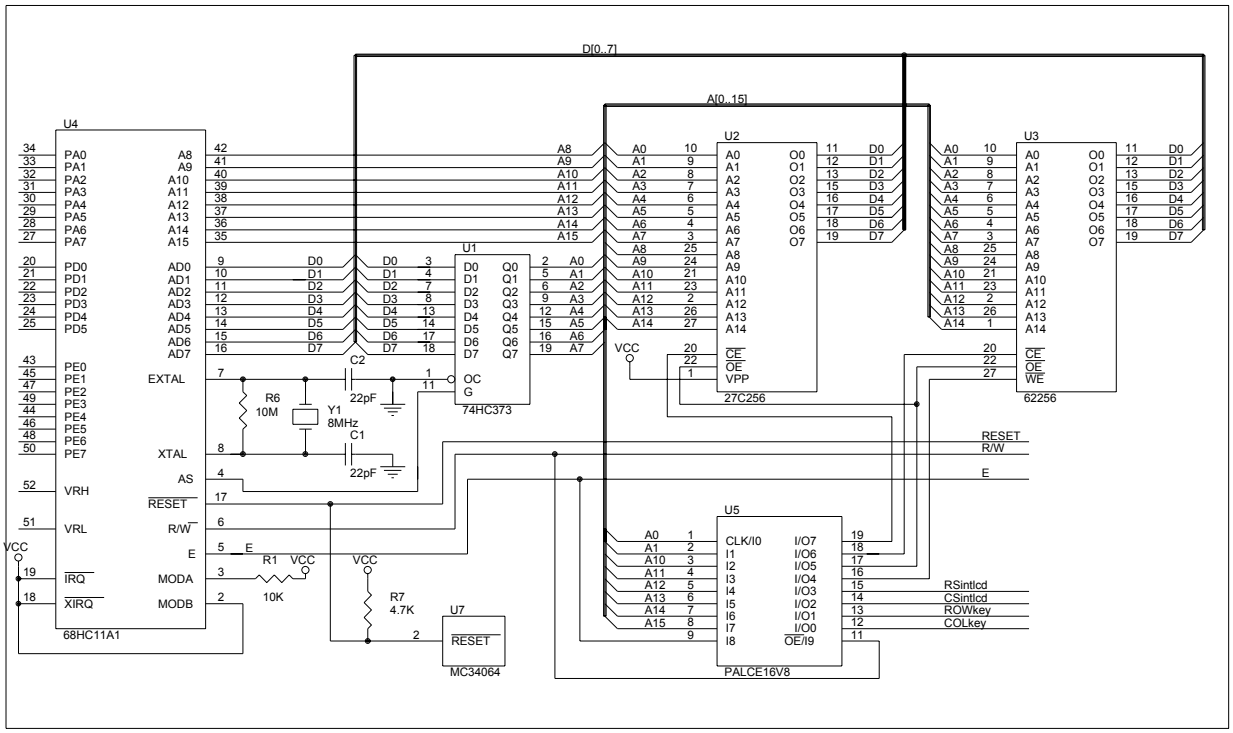


Figura 3-C

3.3 Temporizzazione e scelta della RAM ed EPROM

Occorre adesso analizzare un ciclo di BUS del microcontrollore per poter ricavare così il tempo di accesso massimo che potranno avere sia la RAM che la EPROM. Il 68HC11 può lavorare con una frequenza di clock compresa fra 1 e 3 Mhz per questa applicazione è stato scelta una frequenza media ossia 2 Mhz ottenendo così un periodo di clock pari a 500 ns.

Dato che la parte bassa degli indirizzi è multiplexata con i dati, nel primo semiciclo di clock il MCU genera gli indirizzi e nel secondo genera o legge i dati per cui il primo elemento da considerare per la temporizzazione è il trasparent latches 74HC373.

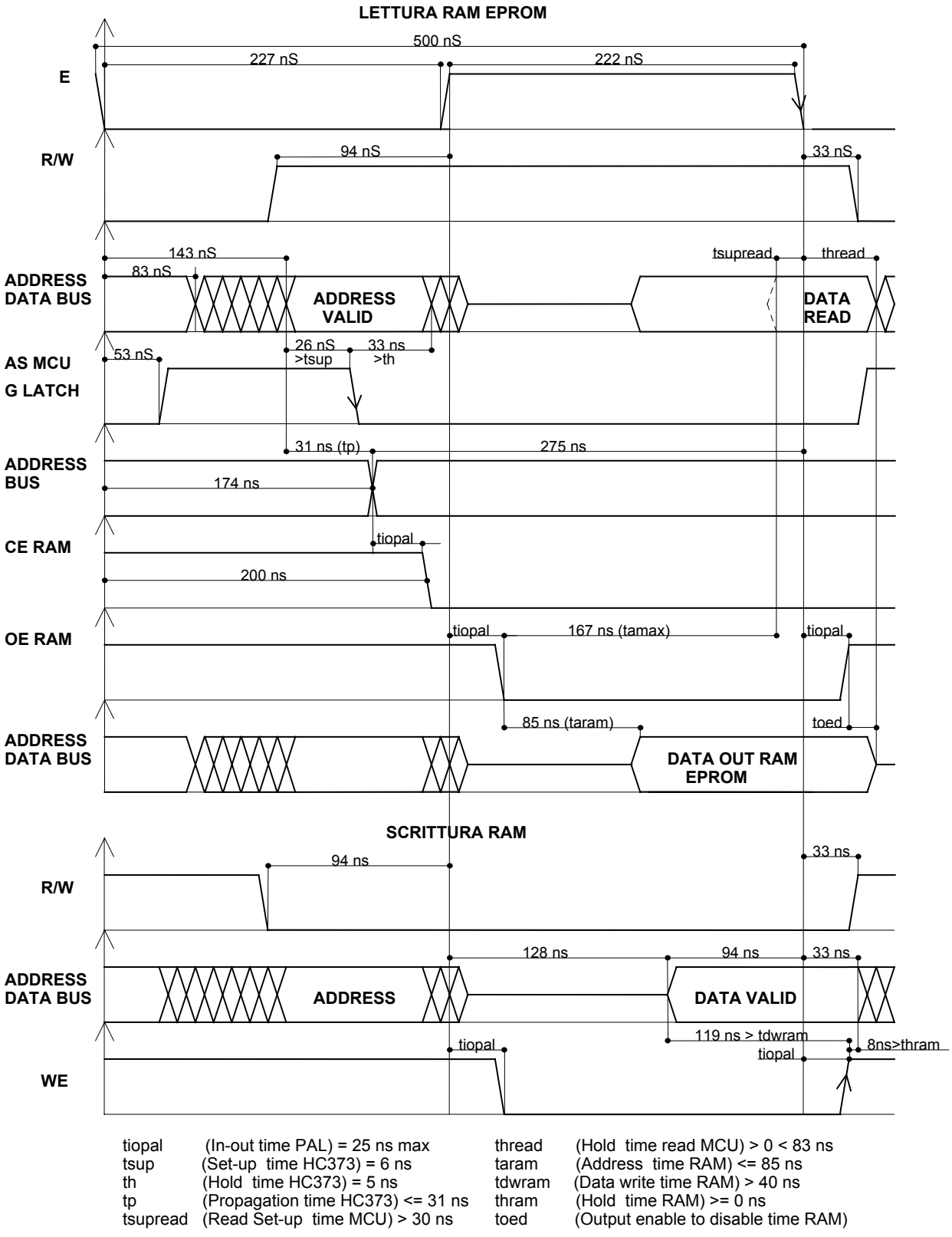
Il 74HC373 viene attivato dal segnale AS, dal momento in cui sono pronti gli indirizzi bassi occorrono 31 ns di tempo di propagazione (tp) del latch affinché gli indirizzi siano trasferiti sul BUS, inoltre affinché vengano memorizzati occorre un tempo di Set-up (tsup)di almeno 6 ns da quando l'MCU genera gli indirizzi al fronte di discesa di AS, ed un tempo di Hold (th) di 5 ns dal momento in cui è sceso il fronte a quando gli indirizzi cambiano, entrambi questi tempi sono ampiamente rispettati infatti il primo tempo vale 26 ns ed il secondo 33 ns.

Gli indirizzi saranno validi sul BUS come si nota in figura 3-4 dopo 174 ns, a questo punto gli indirizzi sono disponibili alla PAL per essere decodificati ed attivare uno dei Chip Enable. La

PAL introdurrà un ulteriore tempo di ritardo dato dai suoi 25 ns di tempo propagazione ingressi-uscite (t_{ioPAL}).

A questo punto è possibile anche attivare l'uscite della RAM o EPROM tramite l'OE, occorre però che sul BUS siano spariti gli indirizzi per evitare una sovrapposizione con i dati, per fare questo basta attivare le uscite, sempre con la PAL, quando il clock E torna alto, in conclusione la memoria sarò attivata 25 ns dopo il fronte di salita del clock.

In fase di lettura della memoria il 68HC11 memorizza i dati sul fronte di discesa del clock è necessario però rispettare un tempo di set-up ($t_{supread}$) di 30 ns mentre non è richiesto tempo di hold thread min=0, come si vede da figura 3-4 il tempo di accesso alla memoria dovrà essere minore di 167 ns (t_{amax})ossia molto elevato rispetto alle memorie in commercio, infatti la scelta è caduta sulla RAM 27C256 e sulla EPROM 62256 entrambe della HITACHI con un tempo di accesso (t_{aRAM}) pari a 85 ns.



tiopal	(In-out time PAL) = 25 ns max	thread	(Hold time read MCU) > 0 < 83 ns
tsup	(Set-up time HC373) = 6 ns	taram	(Address time RAM) <= 85 ns
th	(Hold time HC373) = 5 ns	tdwram	(Data write time RAM) > 40 ns
tp	(Propagation time HC373) <= 31 ns	thram	(Hold time RAM) >= 0 ns
tsupread	(Read Set-up time MCU) > 30 ns	toed	(Output enable to disable time RAM)

Figura 3-D

Adesso occorre verificare che la scelta di questa RAM vada bene anche per la scrittura, infatti questa avviene durante il fronte di salita del segnale WE generato dalla PAL, occorre però che i dati siano mantenuti su BUS prima di tale fronte per 40 ns ($t_{dw\text{-}RAM}$) mentre non è necessario che

si mantengano dopo la salita del fronte ossia $t_{h\text{-}RAM}$=0, come si nota da figura 3-4 entrambi questi tempi sono rispettati il primo vale 94 ns ed il secondo 8ns.

3.4 Progetto interfaccia display LCD:

Prima di analizzare l'interfaccia LCD è necessario analizzare le caratteristiche elettriche-temporali del display.

Dallo schema a blocchi del LMG7400 di figura 3-5 si notano tutti i segnali necessari al pilotaggio; il piedino R/W permette di accedere al display in scrittura o lettura, il piedino RS ha la funzione di distinguere l'accesso all'Instruction Register dall'accesso al Data register, nel primo registro si scriveranno le varie istruzioni di controllo che permetteranno di interpretare la funzione dei dati inviati nel Data Register.

Il piedino E è il segnale di sincronismo quando è alto viene effettuata la lettura mentre sul fronte di discesa si ha la scrittura.

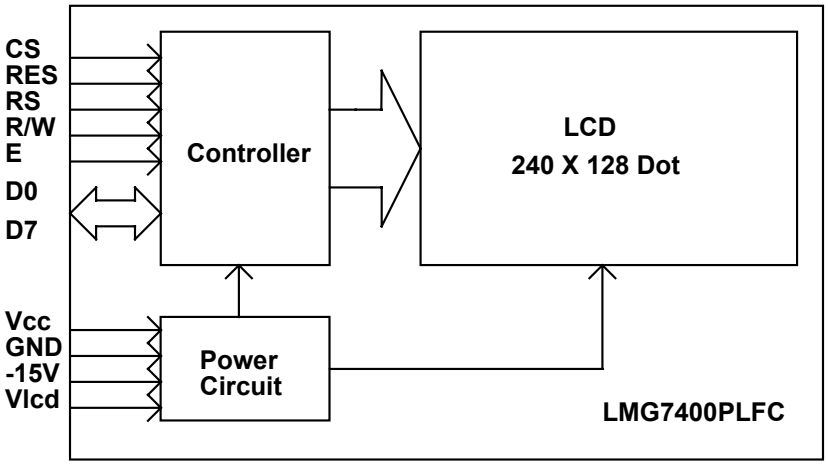


Figura 3-E

Per l'abilitazione si agisce col piedino CS, mentre il pin RES serve per resettare per cui potrà essere collegato direttamente al reset del microcontrollore.

Occorre adesso analizzare le caratteristiche temporali dei segnali, come si nota da figura 3-6 il minimo tempo di ciclo vale 1 us mentre come abbiamo visto il 68HC11 ha un periodo di clock pari a 500 ns, per risolvere questo problema l'idea di base è stata quella di dividere per due il clock al display ed effettuare così ogni volta che si vuol accedere all'LCD una doppia scrittura o doppia lettura, ossia ripetere nel programma due volte la stessa istruzione, la prima volta per dare tempo al display di fornire o acquisire i dati e la seconda per scrivere o leggere effettivamente.

Poiché durante l'intervallo di tempo fra i due accessi viene eseguito il fetch delle istruzioni i segnali generati dall'MCU cambiano stato per cui occorre mantenerli costanti, per i dati in

scrittura si è inserito il latch HC373, mentre per la lettura dei dati si è utilizzato Buffer tree-state HC244, anche i segnali RS e R/W dovranno essere congelati.

Il circuito di figura 3-7 permette di generare i segnali con i giusti limiti temporali, la prima volta che si accede all'indirizzo del display il CSintlcd generato dalla PAL di decodifica fa commutare tramite il flip-flop di tipo Q il segnale di sincronismo Elcd da basso ad alto e si attiva inoltre il display negando Elcd e collegandolo al CS; tramite il trasparent latch inoltre si catturano i segnali RSintlcd e R/W quando E è alto e CSintlcd è basso generando così i segnali RSlcd e R/Wlcd che si manterranno costanti fino alla fine del secondo accesso.

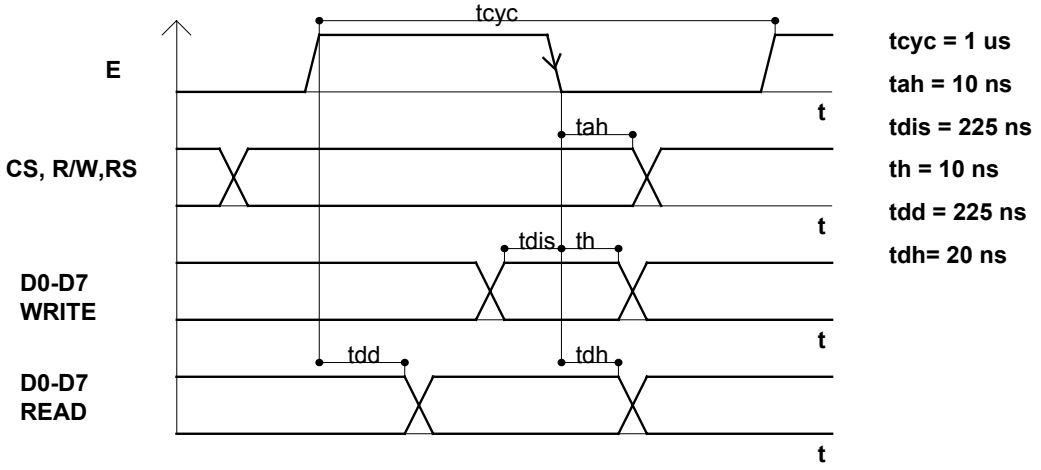


Figura 3-F

Nel caso in cui si effettui una scrittura per far entrare i dati nel latch HC373 occorre che Csintlcd e R/W siano bassi e per essere sicuri di catturare dati anziché indirizzi si tiene abilitato il latch tramite G finche E sta alto, il segnale R/Wlcd abiliterà le uscite del latch mantenendo così i dati stabili al display fino alla fine del secondo accesso. La scrittura nel display avviene quando il segnale Elcd commuta da alto a basso, ossia quando con il secondo accesso il flip-flop D commuterà.

Per la lettura si abilita il buffer tree-state con Csintlcd basso e R/W alto, al primo accesso avremo letto dal display dati non corretti che però non verranno considerati, con la successiva lettura invece il display avrà i dati validi.

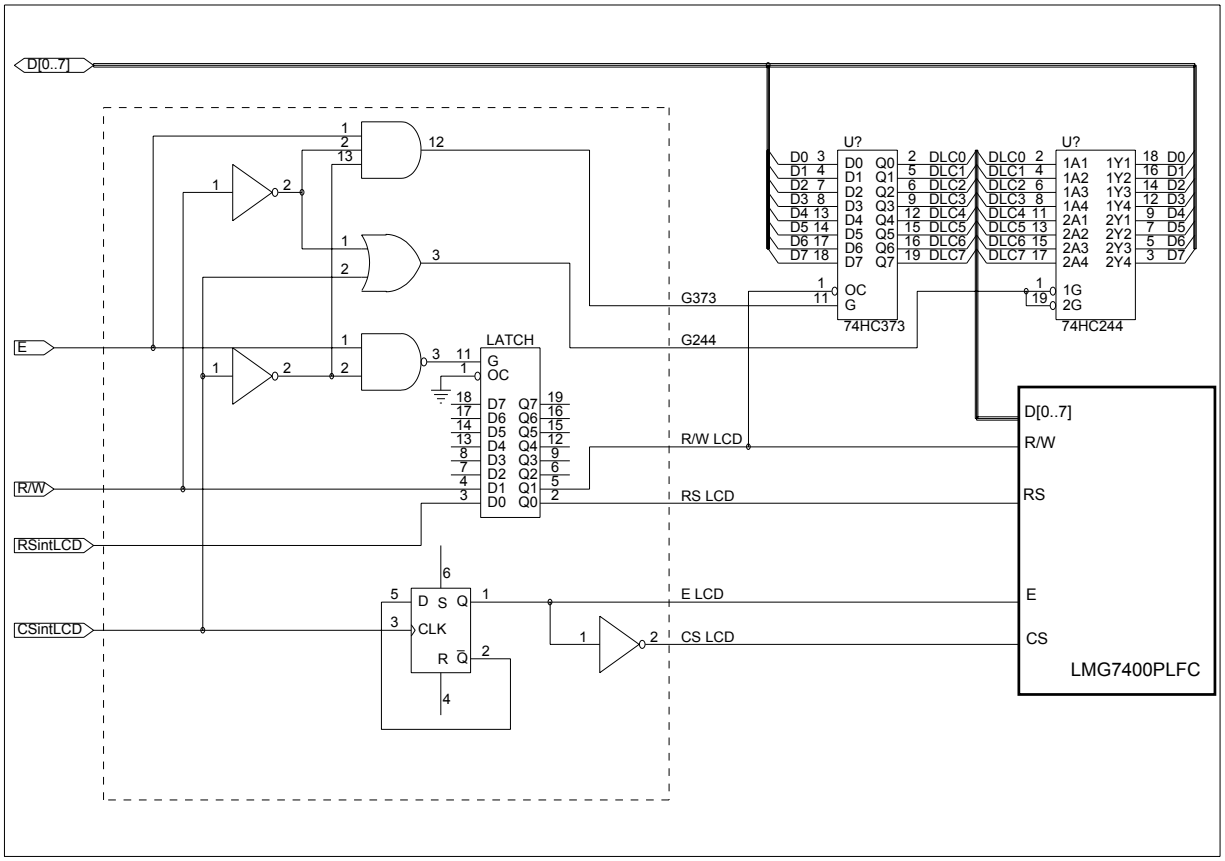


Figura 3-G

Per la parte di schema tratteggiata in figura 3-7 ossia, la logica combinatoria ed il latch per R/W e RS, è risultato vantaggioso utilizzare una PAL anche in questo caso la PALCE16V8, riducendo così tempi di propagazione e molto spazio; l'intera interfaccia LCD sarà per cui realizzata come in figura 3-8.

Il segnale CSintlcd come si osserva dallo schema elettrico di figura 3-8 è stato riportato in ingresso alla PAL al pin Clock/I0 che al pin I1 questo perché tale segnale deve essere trattato sia come ingresso semplice sia come clock per il flip flop D; si nota inoltre che il segnale Elcd torna come ingresso sia al Microcontrollore che alla PAL. Il motivo per cui torna al MCU sta nel fatto che non essendo presente nel flip-flop della PAL un segnale di reset, lo stato iniziale del segnale Elcd sarebbe sconosciuto per cui se ne occuperà il MCU all'accensione di porlo nello stato giusto, mentre il rientro nella PAL è utilizzato, come è spiegato più avanti per generare un ritardo.

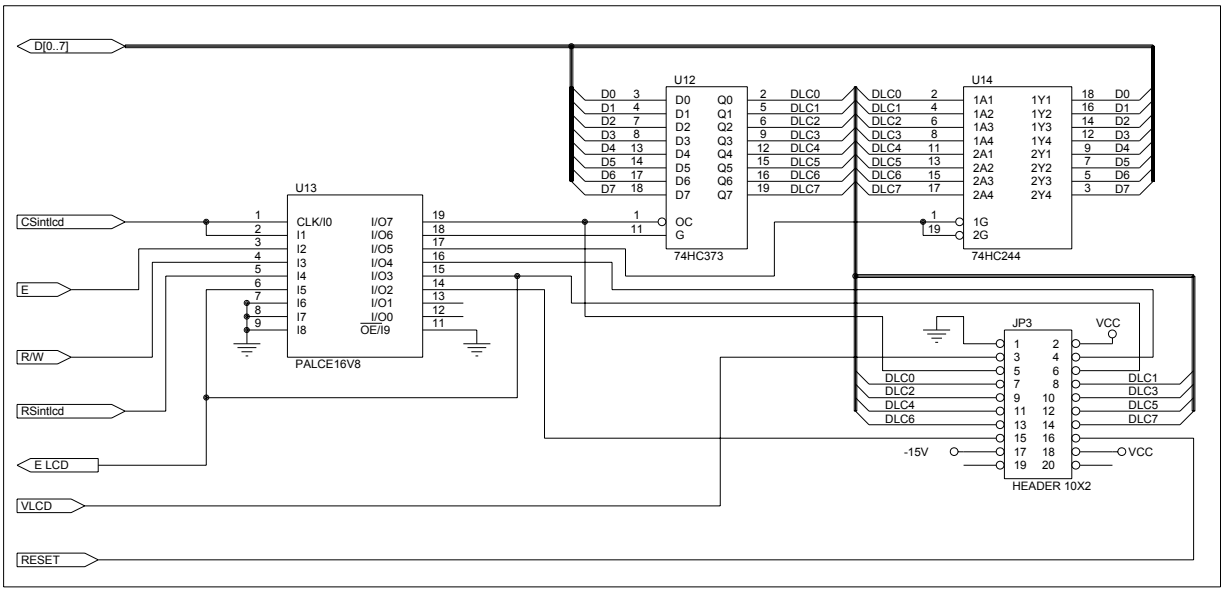
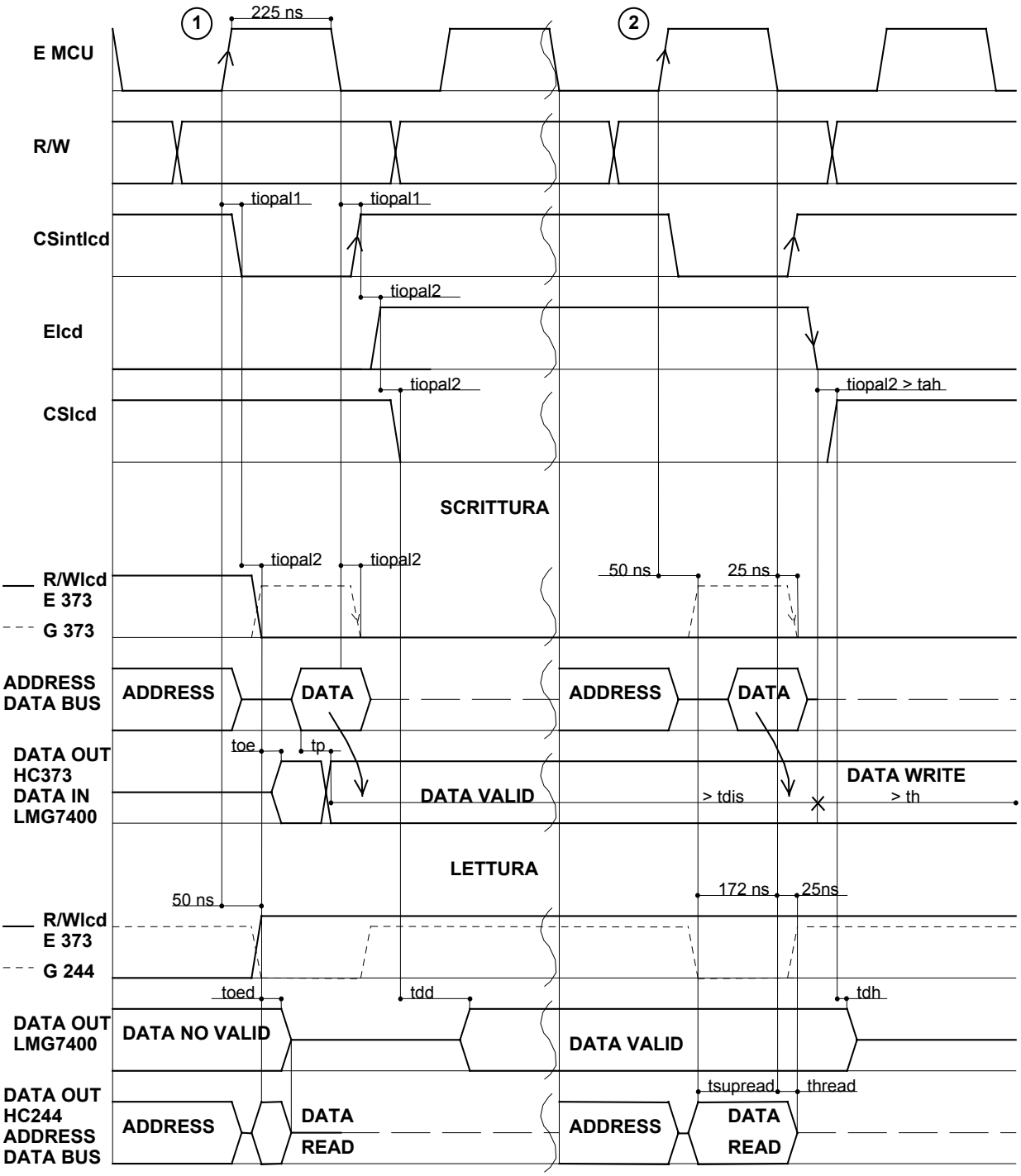


Figura 3-H

Il diagramma temporale di figura 3-9 mostra nei dettagli la temporizzazione dei segnali, tutti i limiti temporali sono rispettati, addirittura alcuni tempi quali il tempo di set up e il tempo di mantenimento in scrittura sono ampiamente superiori al necessario dato che fra i due accessi indicati con 1 e 2 passa più di un ciclo; particolare attenzione è stata dovuta dare però al tempo tah ossia il tempo di mantenimento del segnale CSlcd dal fronte discesa di Elcd pari ad almeno 10 ns, come si vede dallo schema funzionale dell'interfaccia il segnale Cslcd viene ricavato negando Elcd, per creare un ritardo fra i due maggiore di 10 ns anziché generare CSlcd insieme a Elcd si è prima fatto calcolare Elcd poi lo si è riportato in ingresso alla PAL e poi fatto effettuare la negazione generando così un ritardo pari a tiopal sicuramente maggiore di 10 ns.



tiopal	(In-out time PAL) = 25 ns max	toe	(Output enable Time HC373)
tah	(CS Hold time LMG7400)= 10 ns	tp	(Propagatione time HC373)
tdis	(Data Set Up time LMG 7400) = 225 ns	toed	(Output enable to disable time HC373)
th	(Data Hold Time LMG 7400) = 10 ns	tsupread	(Read Set Up time MPU) > 30 ns
tdd	(Data delay time LMG 7400) = 225 ns	thread	(Hold Time read MPU) > 0 <83 ns
tdh	(Data Hold Time LMG 7400) = 20 ns		

Figura 3-I

3.5 Progetto interfaccia tastiera:

Come anticipato la tastiera è stata realizzata tramite una matrice di righe e colonne in cui si attiva una riga alla volta e si va a leggere quale delle colonne è attiva, per implementare una tastiera standard PC-AT sono risultate necessarie 12 righe e 8 colonne ossia una capienza massima di 96 tasti, il numero effettivo di tasti vale invece 90.

Come specifiche per la tastiera oltre al numero di tasti è richiesta anche la decodifica di più pressioni contemporanee, questa caratteristica si è dovuta subito tenere conto in fase di progetto hardware infatti è risultato necessario introdurre per ogni tasto un diodo in serie , come si nota dall'esempio di tastiera in figura 3-10. La funzione di ciascun diodo è quella di impedire che in caso di due o più pressioni di tasti sulla stessa colonna si abbia un ritorno di tensione sulle righe disattive, infatti questo comporta due problemi, il primo è che le righe sono delle uscite di un integrato per cui esso verrebbe danneggiato, il secondo è che essendoci più righe attive allo stesso tempo altri tasti potrebbero portare a livello alto la relativa colonna non potendo così più distinguere a quale riga appartiene il tasto premuto.

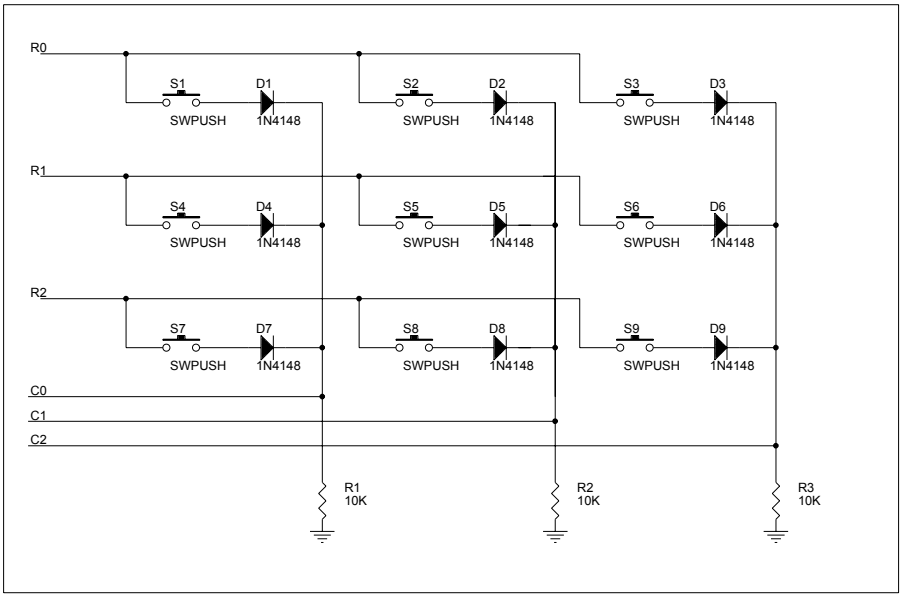


Figura 3-J

Il circuito di interfaccia della tastiera mostrato in figura 3-11 per poter attivare ciascuna riga e mantenerla a livello alto finché non si passa alla successiva è costituito da un latch decoder da 4 a 16 bit, 74HC4514, in pratica via software si incrementa il valore da 0 a 12 e lo si scrive all'indirizzo relativo al latch-decoder mappato in memoria, la PAL attiverà così la scrittura di tale valore nel 74HC4514 portando a livello alto il pin ST.

Per la lettura del byte delle colonne è stato utilizzato il buffer tree-state 74HC244, ogni volta che verrà compiuta una lettura all'indirizzo del buffer nel byte letto sarà attivo il bit relativo alla colonna in cui si trova il tasto premuto ed in base alla riga attiva sarà possibile capire quale dei tasti è stato premuto.

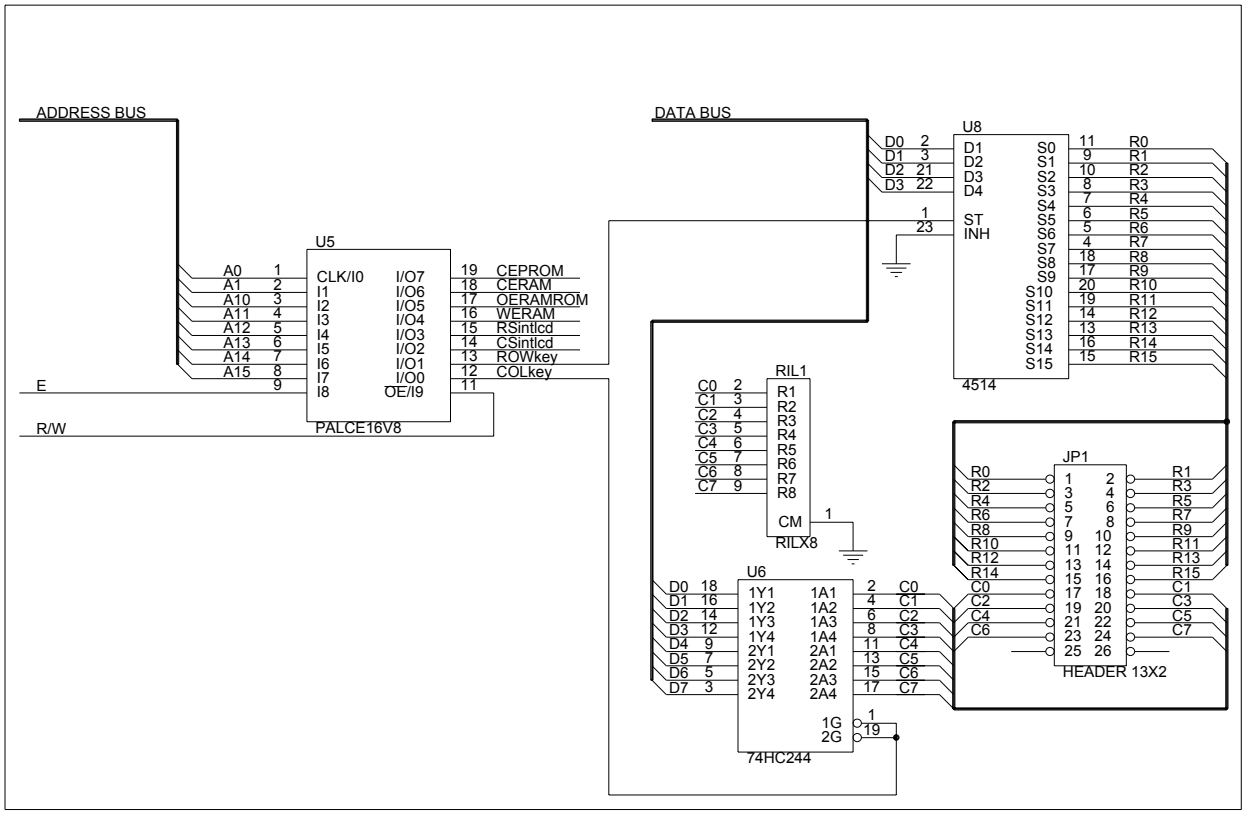


Figura 3-K

3.6 Ingressi analogici:

Come anticipato nel sistema sono presenti due ingressi analogici, un potenziometro e un termistore NTC.

il 68HC11 come ingressi analogici accetta tensioni comprese fra 0 e Vcc, ed ha una risoluzione di 8bit, occorre però fissare i limiti di inizio e fondo scala tramite i pin VRH e VRL, la tensione applicata al pin VRL sarà la tensione in ingresso analogico alla quale corrisponde il valore

digitale 0, mentre la tensione al VRH sarà la tensione analogica alla quale corrisponde il valore digitale 255.

Per fare in modo di essere sicuri di poter raggiungere i limiti di inizio e fondo scala i valori di VRL e VRH sono stati fissati rispettivamente di 0.05 V e 4.95 V tramite un partitore resistivo come si nota da figura 3-12.

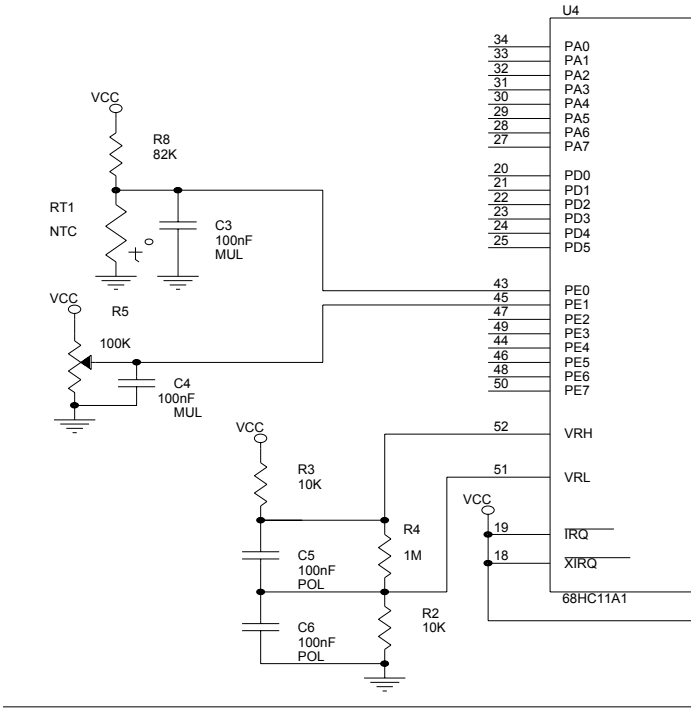


Figura 3-L

Il potenziometro viene utilizzato per una regolazione manuale del contrasto del display per cui non c'è nessun accorgimento da tener conto, il termistore invece ha la funzione di compensare la variazione delle tensione di regolazione contrasto rispetto alla temperatura per cui è necessario che il legame fra queste due grandezze sia lineare oltre che preciso. Si è dovuto affrontare allora il problema che sia l'andamento della resistenza del termistore che la tensione Vlcd da applicare al display non sono lineari rispetto alla temperatura.

Per cercare di linearizzare il tutto il metodo poteva essere o via hardware o via software, in questo caso è risultato molto comodo agire via hardware infatti scegliendo adeguatamente il valore della resistenza di polarizzazione dell'NTC, si è riuscito a rendere lineare il tutto.

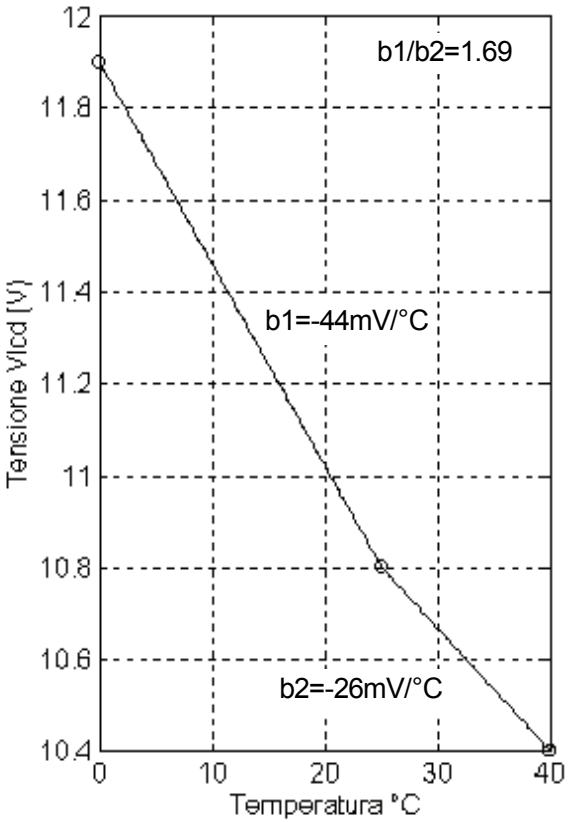
Le caratteristiche elettriche del display riportano per la tensione di pilotaggio display 3 valori come si osserva da tabella, nel grafico 1 di figura 3-13 è rappresentata la curva interpolante i tre

punti in valori assoluti, come si osserva non è lineare ma forma una spezzata, con una pendenza di -44 mV/°C al di sotto di 25°C ed una di -26 mV/°C fra 25 e 40°C.

Ta	Vlcd
0°C	-11.9 V
25°C	-10.8 V
40°C	-10.4 V

Il termistore scelto presenta un valore di resistenza nominale (T=25°C) di 22kohm, la caratteristica temperatura-resistenza è riportata nel grafico 2 di figura 3-13, dato che per il display sono forniti solo i tre valori di tensioni per le temperature di 0, 25 e 40°C è risultato sufficiente interpolare la curva dell'NTC per questi 3 valori approssimandola con due rette come si osserva sempre da figura 3-13. Facendo un confronto con la caratteristica di tensione Vlcd che deve avere il display, si osserva che entrambe le curve hanno più o meno lo stesso andamento, mentre per la prima il rapporto fra le due pendenze a1/a2 vale 2.72, per la seconda il rapporto vale b1/b2 ossia 1.69, dunque l'azione di correzione sta nell'avvicinare il più possibile questi due rapporti.

1) Tensione LCD raccomandata



2) Caratteristica Temperatura-Resistenza

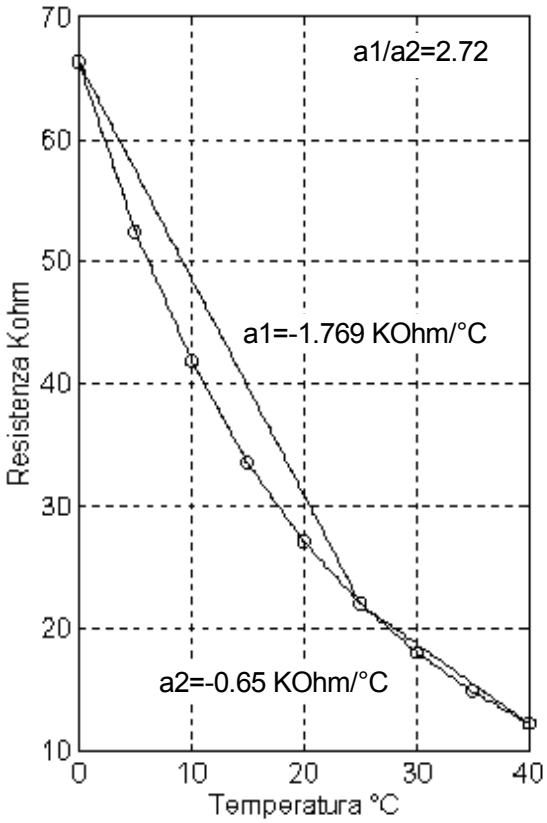


Figura 3-M

La funzione che approssima con due rette la caratteristica del termistore è la seguente:

$$Rt(t) = \begin{cases} Rt_1(t) = -1.769t + 66.24 \\ Rt_2(t) = -0.65(t-25) + 22 \end{cases} \text{ per } \begin{matrix} t \le 25°C \\ t \ge 25°C \end{matrix}$$ con Rt(t) espresso in Kohm, e t in gradi centigradi, la tensione data dal partitore sarà di conseguenza espressa come insieme di due funzioni ottenendo così:

$$V_O(t) = \frac{Rt\ (t)}{Rt\ (t) + R8} = \begin{cases} V_{O1}(t) = V_{CC}\dfrac{Rt_1(t)}{Rt_1(t) + R8} \\ V_{O2}(t) = V_{CC}\dfrac{Rt_2(t)}{Rt_2(t) + R8} \end{cases} \text{ per } \begin{matrix} t \le 25°C \\ t \ge 25°C \end{matrix}$$

Affinché la tensione uscente dal partitore abbia lo stesso andamento di quella richiesta dal display, si impone che il rapporto fra le pendenze delle due curve calcolate negli estremi 0, 25 , 40°C sia pari al rapporto fra i coefficienti angolari b1/b2, occorre per cui trovare il valore di R8 affinché valga la seguente condizione:

$$\frac{\dfrac{\Delta V_{O1}(t)}{\Delta t}}{\dfrac{\Delta V_{O2}(t)}{\Delta t}} = 1.69 \text{ ,da cui : } \frac{\dfrac{V_{O1}(25) - V_{O1}(0)}{25}}{\dfrac{V_{O2}(40) - V_{O2}(25)}{15}} = 1.69$$, sostituendo i valori e risolvendo l'equazione si ottiene un valore di resistenza pari a 77.41 Kohm, il valore commerciale più prossimo è di 82 KOhm.

Con tale valore di resistenza come si vede da tabella i coefficienti delle due rette danno un rapporto di 1.712 molto prossimo all'ideale di 1,69 più che sufficiente data la tolleranza dei componenti, inoltre per quanto riguarda l'escursione di tensione benché non sia molto elevata circa 1,6 Volts, con la risoluzione a disposizione di 20 mV si distinguono 81 valori di tensione che bastano per questo tipo di regolazione.

T°C	Vo (V)	ΔVo/ΔT mV/°C	c1/c2
0	2.234	c1=-9.42	1.712
25	1.057	c2=-5.5	
40	0.645		

3.7 Progetto alimentatore e integratore PWM:

L'intero sistema necessita di varie tensioni di alimentazione, per la parte digitale di controllo che è interamente CMOS sono necessari 5V, per il display invece occorrono varie tensioni, 5V per la parte logica, 12 V per l'alimentatore della lampada CFL, -15V per il circuito driver dei cristalli liquidi, e una tensione compresa fra -10.4 e -11.9 come anticipato per la regolazione del contrasto.

La tensione di alimentazione in ingresso vale 24 V con una tolleranza di +20 % e -10% , ossia compresa fra 21.6 V e 28.8 V. Per generare le tensioni negative e le altre tensioni è risultato vantaggioso realizzare un convertitore DC/DC utilizzando il regolatore switching MC34167 della Motorola che necessita di un minimo numero di componenti esterni e contiene al suo interno il generatore PWM, l'integrato può lavorare con tensioni di ingresso fino a 40V ed è in grado di fornire corrente continua fino a 5A, più che sufficiente dato il basso consumo della tecnologia CMOS.

Le altre caratteristiche di questo regolatore sono:

- Frequenza di oscillazione fissa di 72 kHz
- Riferimento di tensione interno di 5.05 V
- Duty Cycle variabile fra 0% e 95 %
- Precisione rispetto al riferimento di tensione del 2%
- Limitazione di corrente ciclo per ciclo
- Compensazione interna della temperatura

Come si osserva dallo schema elettrico di figura 3-14 il regolatore MC34167 è realizzato con case a 5 piedini: la tensione d'ingresso viene applicata fra Vin e Rgnd, il piedino Swout è l'uscita switching, il pin Sense è l'ingresso per la reazione, il pin Comp, collegato tramite un filtro RC all'ingresso Sense, permette di effettuare una compensazione che stabilizza l'anello di reazione

Dato che le tensioni positive sono due una di 12V e l'altra di 5V, si è utilizzato l'alimentatore switching per generare il 12V, poi si è utilizzata questa tensione per generare la Vcc di 5V tramite il regolatore 78L05 , mentre per le tensioni negative si è sfruttato direttamente l'uscita switching Swout del regolatore come spiegato più avanti.

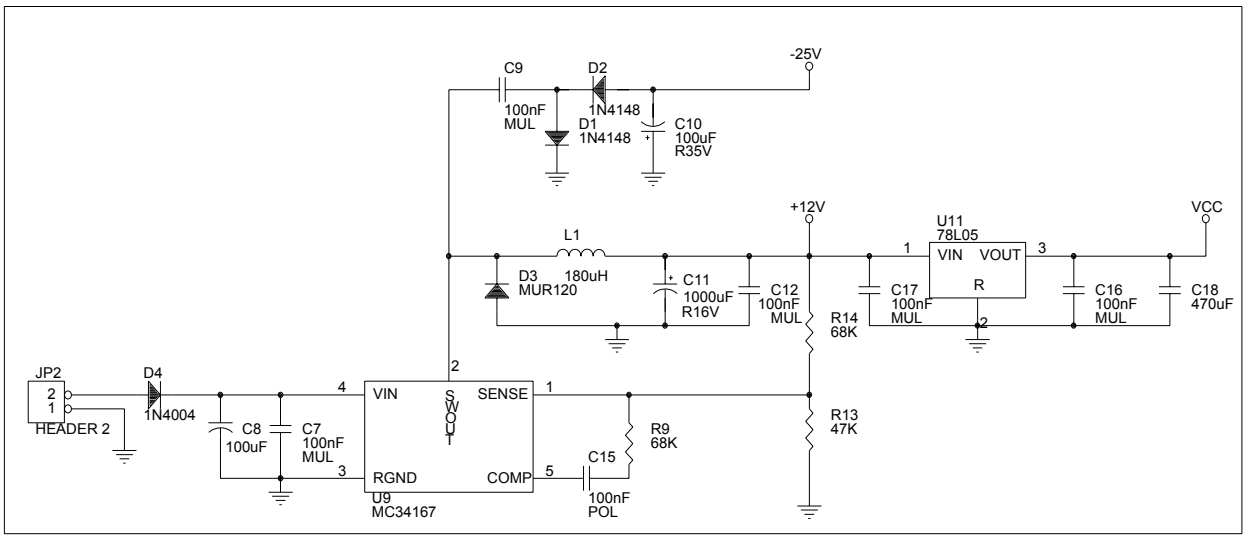


Figura 3-N

Dallo schema elettrico si osserva la configurazione tipica step down, nel progetto classico di questo tipo di convertitore DC/DC si considera come specifica il minimo assorbimento di corrente in quanto la corrente sull'induttanza, come si osserva in figura 3-15, varia linearmente entro due limiti, alla fine della carica raggiunge I_{Lmax} e alla fine della scarica $I_{lmin,}$ dato che la

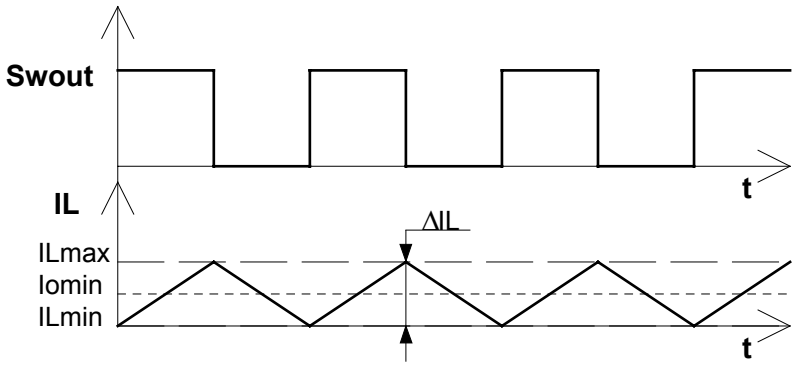


Figura 3-O

corrente di uscita I_O è uguale alla loro media, occorre imporre che nel caso di minima corrente assorbita dal carico, la I_{Lmin} valga al minimo 0, ossia ΔI_L sia al massimo doppia della Iomin, occorre allora dimensionare l'induttanza affinché si rispetti la relazione $\Delta I_L \leq 2 I_{O\,\min}$.

Questa condizione rappresenta il limite fra funzionamento continuo e discontinuo, infatti se la corrente di uscita risulta essere minore di quella minima di specifica, la corrente sull'induttanza

,durante il periodo toff, dovrebbe diventare negativa per riuscire a mantenere la media pari a Io, in realtà la corrente si ferma a 0 e vi rimane finché non inizia la fase di carica, in questa situazione si ha il funzionamento discontinuo del convertitore.

La relazione che lega ΔI_L al valore dell'induttanza è la seguente: $\Delta I_L = \frac{V_o(V_i - V_o)}{fLV_i}$, dove V_o è la tensione di uscita, Vi la tensione di ingresso, ed f è la frequenza di lavoro, come si osserva la variazione di corrente su l'induttanza è inversamente proporzionale ad L, per cui se si volesse garantire la condizione di funzionamento continuo del convertitore il valore dell'induttanza dovrà essere tanto più grande quanto più piccola è la corrente assorbita.

Il consumo minimo del terminale si aggira intorno agli 60 mA, supponendo di rispettare la condizione di funzionamento continuo, e considerando il caso peggiore con la tensione di ingresso massima di 28.8 V, sostituendo i valori nella formula inversa $L = \frac{V_o(V_i - V_o)}{f 2 * I_{O\min} V_i}$ si otterrebbe un valore di induttanza pari a $L = \frac{12(28.8-12)}{72 \cdot 10^3 \cdot 2 \cdot 60 \cdot 10^{-3} \cdot 28.8} = 810\mu H$, il valore commerciale più prossimo sarebbe di 1mH, un induttanza così andrebbe ad occupare molto spazio, così per risparmiare sia spazio che denaro si è preferito far lavorare il convertitore in modo discontinuo scegliendo l'induttanza L1 pari a 180μH. Tale scelta comporta ulteriori vantaggi, infatti nel funzionamento discontinuo il convertitore ha una risposta alle brusche variazioni della tensione in ingresso molto più rapida, dato che essendoci meno energia accumulata nell'induttanza c'è anche meno inerzia, l'unico svantaggio è che si hanno dei picchi di corrente superiori nel diodo e nel condensatore che sono comunque sopportati, infatti il diodo D3 MUR120 ha una corrente massima di 50 A.

Per il calcolo del condensatore di filtro C11 si è imposto un ripple sull'uscita di al massimo 2 mV utilizzando la seguente formula $C = \frac{V_o(V_i - V_o)}{8 f^2 L V_i \Delta V_o}$ si ricava un valore minimo di C pari a 395 μF, considerando che la tolleranza dei condensatori elettrolitici può raggiungere anche il 40 %, il valore commerciale più adatto e anche più diffuso è risultato il 1000 μF.

La formula che lega la tensione di uscita al rapporto delle resistenze del partitore resistivo è la seguente $V_0 = 5.05(1 + \frac{R14}{R13})$, il 5.05 che compare è il riferimento di tensione che il regolatore possiede al suo interno, per ottenere una tensione Vo pari a 12 V il rapporto fra le due resistenze deve valere 1.376, i valori commerciali delle resistenze che si avvicinano di più a questo rapporto sono per R14 68KΩ e per R13 47KΩ, infatti con una tolleranza del 5% il loro rapporto

varia fra $\frac{68-5\%}{47+5\%}=1,31$e $\frac{68+5\%}{47-5\%}=1.59$, che sulla tensione di uscita comporta un limite minimo di tensione di 11.61 V e un limite massimo di 13.12 V, limiti accettabili dato che questa tensione viene applicata direttamente solo all'alimentatore per la lampada CFL.

Per generare la tensione negativa di partenza si è sfruttato direttamente l'uscita Swout del regolatore, applicandola al circuito formato dai due diodi D1 e D2 e dai condensatori C9 e C10.

Come si osserva dai diagrammi di figura 3-16 quando Swout è alto il condensatore C9 si carica molto rapidamente data la sua bassa capacità e la bassa resistenza di conduzione del diodo D1, raggiungendo una tensione pari a Vswh-Vdsat; quando Swout commuta a 0 il condensatore per mantenere la sua tensione di carica sarà costretto a portare la Vd a -Vswh + Vdsat, a questo

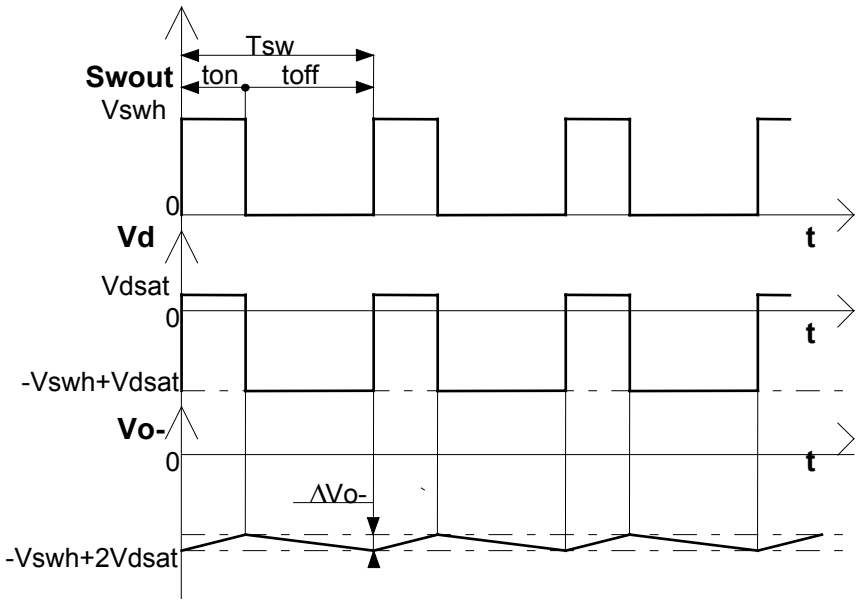


Figura 3-P

punto il diodo D2 può entrare in conduzione e caricare il condensatore C10 con una tensione negativa pari al massimo a -Vswh + 2 Vdsat, considerando Vswh circa uguale a Vin ossia 24V e Vdsat circa 0.7 la Vo- raggiunge i -22.6V.

Per quanto riguarda il dimensionamento del condensatore C10 occorre conoscere la massima corrente assorbita dal carico, e il massimo tempo di scarica del condensatore ossia toffmax, dato che considerando la scarica lineare la variazione di tensione all'uscita è pari a $\Delta V_{O-}=\frac{I_{O-}\cdot t_{Off}}{C10}$.

Per il calcolo di t_{offmax} occorre conoscere il Duty-Cicle minimo, questo caso si verifica quando Vin è massima e Vo è minima dato che la tensione di uscita cale $V_O=V_{In}\cdot\frac{t_{on}}{t_{off}}$, sostituendo si

ottiene un valore di Duty Cicle pari a $\frac{t_{on}}{t_{off}} = \frac{28.8}{11.61} \cong 0.4$ e dato che il periodo vale $T = \frac{1}{f} = \frac{1}{72 \cdot 10^6 Hz} \cong 14\mu s$, il valore di toffmax varrà $T \cdot (1 - 0.4) = 8.4\mu s$.

La corrente che deve fornire questa uscita dipende solo dal display il quale indica un assorbimento totale per le tensioni negative pari a circa 10 mA, il ripple sulla tensione lo si è imposto di circa 1 mV, con tali valori si ottiene un valore di capacità pari a $C = \frac{I_{o-\max} \cdot t_{off\,\max}}{\Delta V_O} = \frac{10 \cdot 10^{-3} A \cdot 8.4 \cdot 10^{-6} s}{10^{-3} V} = 84\mu F$, ossia il valore commerciale più prossimo di 100μF.

Tramite questa tensione negativa adesso è possibile ricavare facilmente le tensioni da applicare al display, infatti per generare i -15 V per la Vee basta moltiplicare per tre la Vcc di 5V tramite un semplice operazionale, alimentato fra Vcc e Vo-, l'altra tensione negativa da inviare al display è quella di regolazione contrasto, per questa è necessario realizzare un blocco integratore per il segnale PWM uscente dal micro compreso fra 0 e 5V, e poi invertire di segno e amplificare il livello di tensione costante uscente dall'integratore in modo da rispettare i limiti di tensione sul display.

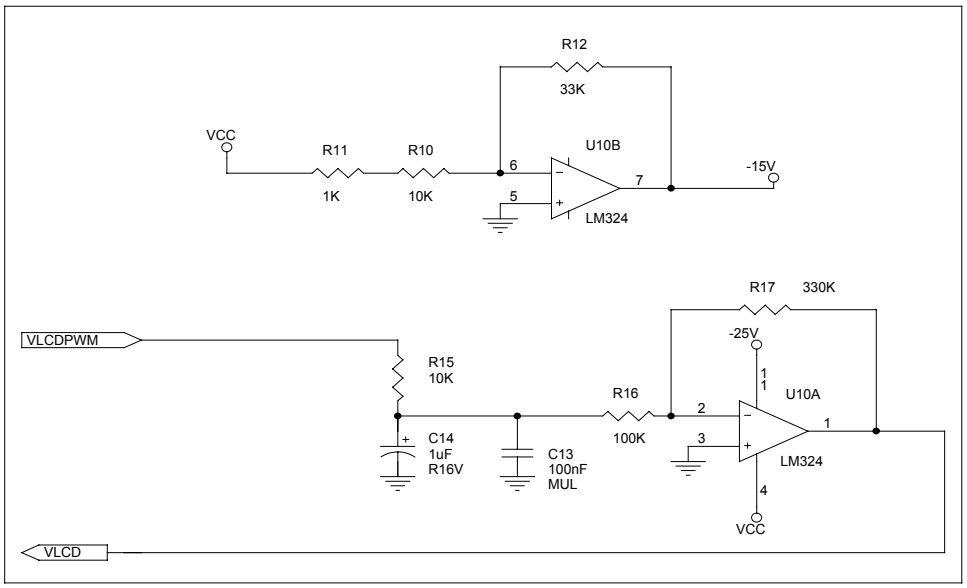


Figura 3-Q

Il circuito che si occupa di integrare il segnale PWM, è costituito come si osserva da figura 3-17 da un filtro RC costituito da R15 e C14, il valore della costante di tempo si è imposto di 0.1 s, molto elevato rispetto al periodo del PWM di 0.1 ms in modo che le variazioni di tensioni siano

graduali, inoltre si è posto anche il condensatore C13 di tipo ceramico per filtrare i disturbi a frequenza elevata generati dal micro.

Dato che la tensione integrata deve anche essere amplificata e invertita, in cascata al filtro RC si è posto un amplificatore invertente il cui guadagno permette di fornire il giusto range di tensione in ingresso al display, il guadagno deve essere tale da fornire con il livello massimo in ingresso di 5 V ossia Duty-Cicle 100% il valore massimo negativo di tensione del display ossia circa -12 V, cioè un guadagno di almeno -2.4, per cui affinché l'amplificatore invertente abbia questo guadagno occorre avere un rapporto fra R17 e R15+R16 proprio pari a 2.4, utilizzando i valori di resistenza riportati nello schema aventi tolleranza del 5% si ottiene un guadagno che può variare fra 2.7 e 3.3, sufficiente per cui a generare la massima tensione negativa, e non troppo elevato da danneggiare il display infatti la massima tensione che può uscire al limite vale $V_{lcd\max} = 3.3 \cdot 5V = 16.5V$ e la massima sopportata vale 16.8 V.

4. PROGETTAZIONE SOFTWARE

4.1 Descrizione ambiente di sviluppo e organizzazione per la programmazione:

Per lo sviluppo del software è stato utilizzato un normale PC, per scrivere il programma, assemblarlo e linkarlo, per provare via via il corretto funzionamento del programma è risultata indispensabile una scheda emulatore realizzata dalla casa costruttrice MOTOROLA la quale permette di sostituirsi al microcontrollore 68HC11, ed alla memoria EPROM della scheda, offrendo così la possibilità di provare il programma con un debug e di modificarne in tempo reale il contenuto, riducendo così notevolmente i tempi di sviluppo.

Per dividere logicamente le varie parti del programma, si è utilizzato una programmazione a moduli, il modulo principale contenuto nel file *MAIN.ASM* oltre a contenere l'indirizzo iniziale del programma (*entry point*) e le varie inizializzazioni delle variabili in RAM, contiene il ciclo principale (*main loop*) in cui gira continuamente il programma che andando a richiamare le varie routine contenenute nel muduli secondari, permette ad ogni ciclo di osservare lo stato del sistema ed agire di conseguenza.

I moduli secondari sono i seguenti sei:

1) *SUBG.ASM,* questo modulo contiene tutte le varie Subroutine utilizzate dagli altri moduli, routine comunque ad alto livello , cioè che non lavorano direttamente con le interfacce esterne.
2) *PERI.ASM,* questo modulo invece contiene tutte le routine di inizializzazione e gestione delle periferiche esterne.
3) *VT100.ASM,* in questo modulo sono contenute tutte le routine per l'emulazione dello standard VT100 e VT52, sia per la ricezione che per la trasmissione.
4) *TAB.ASM,* questo modulo non contiene nessuna routine, ma solo tabelle utilizzate dal resto del programma, come tabelle di conversione, ad esempio per l'interpolazione della tastiera, tabelle contenenti indirizzi di subroutine per rendere più chiaro il programma.
5) *VETT.ASM,* questo modulo contiene l'Interrupt Vector ossia una tabella che associa ad ogni interrupt la label della routine di servizio da eseguire, questa tabella sarà poi posizionata all'indirizzo $FFC0.
6) *EQUMAC.ASM,* questo file in realtà non è un vero e proprio modulo, infatti contiene tutte le *Equates*, ossia associazioni mnemoniche con indirizzi o flags, e *Macro* ossia routine di più alto livello utilizzate dagli altri moduli.

Per la programmazione è indispensabile anche un altro file, *68HC11.REG* in questo file sono definite le associazioni fra registri del micro ed il loro indirizzo, in modo da poter lavorare con nomi simbolici, anziché con numeri.

4.2 Organizzazione del software:

Da figura 4-1 si osserva come è strutturato il software da un punto di vista funzionale, in pratica il programma si avvia eseguendo una prima fase di inizializzazione, in cui si imposta il funzionamento del MCU, e si inizializzano le variabili in memoria e le periferiche sia interne che esterne, in questa fase il programma passerà non solo alla prima accensione, ma ogni volta che si genera un reset.

Dopo questa prima fase si entra nel *main loop*, all'interno di questo anello si svolge il vero e proprio programma, in pratica ad ogni ciclo si leggeranno i vari ingressi, si testeranno le periferiche, ed in base ai dati letti si eseguiranno le varie parti di programma relative.

Il main loop e le routines che esso può aver chiamato possono essere interrotti ogni volta che si scaturisce un interrupt, infatti in tal caso verrà eseguita la routine di interrupt relativa ed una volta terminata il micro tornerà al punto dove era stato interrotto.

Le operazioni principali che si eseguono durante il Main loop sono: leggere il valore del potenziometro e del sensore di temperatura tramite l'AD converter, ed aggiornare così la variabile che regola il duty cicle del PWM, vedere se si sono ricevuti caratteri da seriale, e se si allora eseguire l'azione relativa al carattere ad esempio stamparlo sullo schermo, infine scandire la tastiera per vedere se c'è un tasto premuto, e se c'è eseguire l'azione relativa, ad esempio spedire il carattere corrispondente su seriale.

Si ottiene così un diagramma di flusso come in figura 4-2.

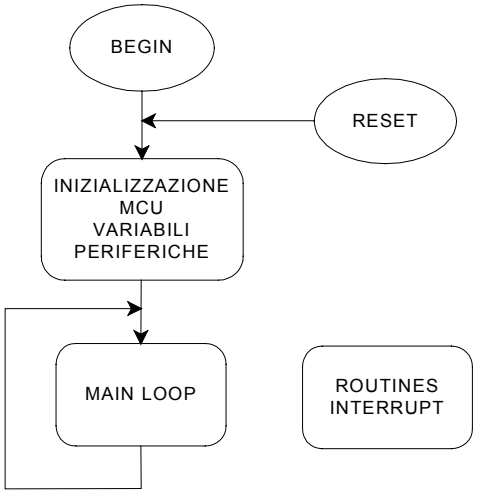


Figura 4-A

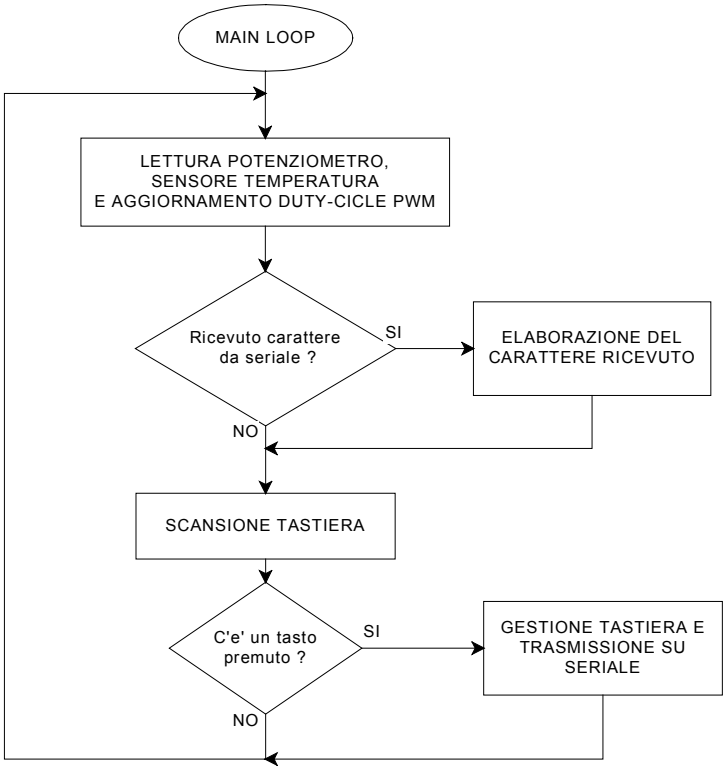


Figura 4-B

4.3 Routines di basso livello

Vista l'impostazione generale del programma occorre adesso analizzare le parti di programma a più basso livello, ossia le parti di interfacciamento Hardware, quali la generazione del segnale PWM, la scrittura su display LCD, la gestione della comunicazione seriale e la scansione della tastiera.

4.3.1 Generazione del segnale PWM:

Il segnale PWM da generare è un segnale periodico con una frequenza di circa 10 Khz, il Timer interno del 68HC11 permette con facilità di generare eventi periodici, infatti è costituito da un contatore FREE RUNNING, ossia un contatore libero che non può essere bloccato, e contiene inoltre dei registri chiamati TOC? (Timer Output Compare) che grazie ad un comparatore permettono di essere confrontati con il valore attuale del contatore FREE RUNNING, quando si verifica un'uguaglianza si possono effettuare varie azioni direttamente sui Pin di uscita relativi.

Per generare un'onda quadra con duty-cicle variabile la tecnica è quella di utilizzare due registri di Output-compare, uno per portare l'uscita alta ogni volta che è trascorso il periodo TPWM, ed un altro registro che porta l'uscita bassa ogni volta che è trascorso il THPWM ossia il tempo in cui l'uscita rimane alta, regolando così la variabile THPWM è possibile variare il duty-cicle.

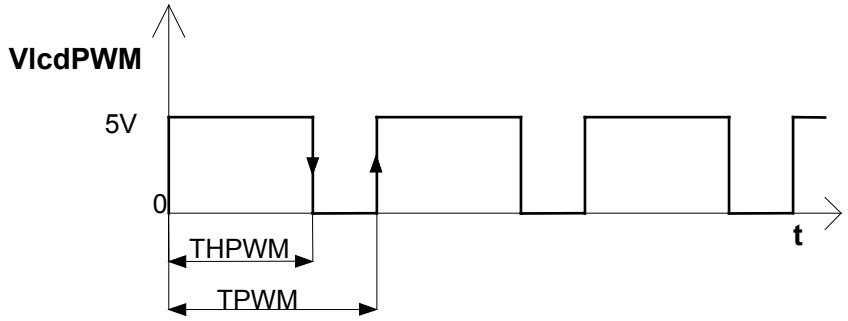


Figura 4-C

Nello schema di figura 4-4 è riportata la parte di Timer interessata per la generazione del segnale PWM, nel registro TOC1 viene inserito la costante TPWM , nel registro TOC2 la variabile THPWM, ogni volta che si verifica un uguaglianza fra contatore FREE RUNNING e registri vengono settati i flags di stato OC1F o OC2F che rispettivamente commuteranno l'uscita PA6 alta oppure bassa, infatti come si osserva il flag OC2F agisce direttamente sull'uscita PA6, mentre per OC1F è possibile indicare su quale pin della porta di uscita agire, in tal caso anch'esso su PA6.

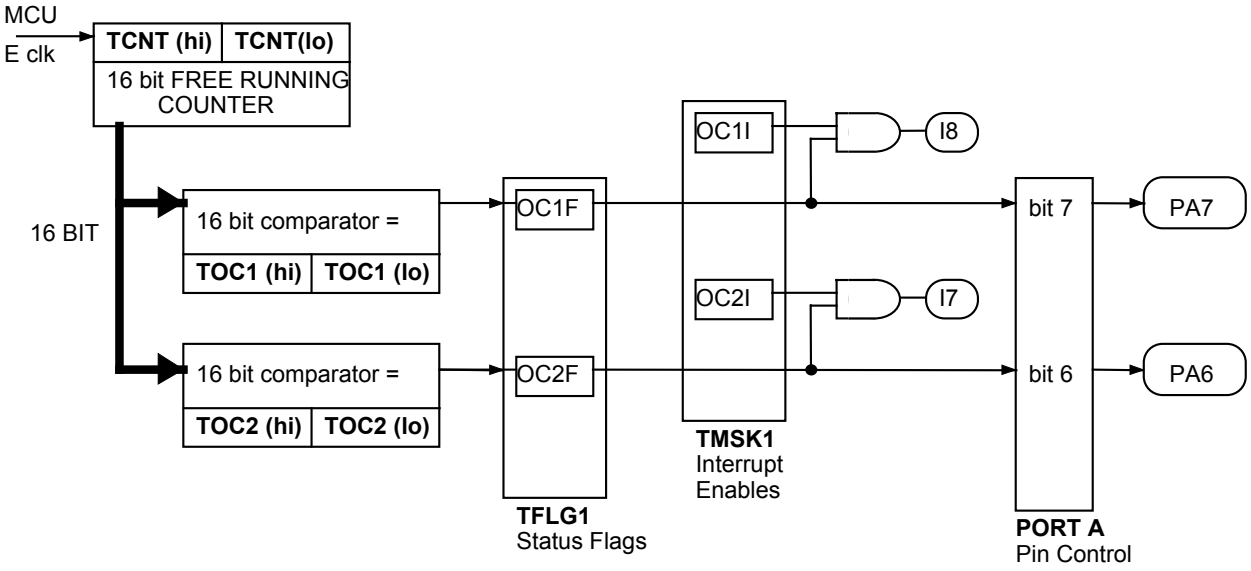


Figura 4-D

Il contatore FREE RUNNING viene incrementato direttamente dal Clock del MCU di frequenza pari a 2 Mhz, per portare l'uscita alta con una frequenza di circa 10 KHz si è imposto TPWM pari a 255 decimale con questo valore si ottiene una frequenza pari a 2Mhz/256 ossia 7812 Hz,

poiché il contatore conta fino a 2^{16} ossia 65535 occorre aggiornare i registri di compare TOC1 e TOC2 ogni volta che termina il periodo TPWM, altrimenti non si verificano le uguaglianze finché il contatore non riparte da 0, per fare questo aggiornamento il modo migliore è stato farlo sotto interrupt, infatti il timer offre il registro TMSK1 con il quale è possibile in corrispondenza ad un uguaglianza generare anche un interrupt relativo al registro di compare specificato, nel nostro caso abilitando OC1I si genera un interrupt ogni volta che è trascorso il periodo TPWM. La routine di interrupt lanciata leggerà il valore attuale del ragistro TOC1 vi sommerà la variabile THPWM e salverà questo valore nel registro TOC2, inoltre incrementerà TOC1 di TPWM, ovviamente questa routine di interrupt verrà eseguita con una frequenza pari a quella del segnale PWM ossia 7812 Hz.
La routine di inizializzazione *INIPWM* e la routine di interrupt *GENPWM* per la generazione del segnale PWM si trovano nel modulo INT.ASM.

4.3.2 Impostazioni del display LCD e scrittura:

Per poter utilizzare il display LCD la prima cosa da fare è inizializzare lo stato del segnale di sincronismo ELCD che, come anticipato nella descrizione hardware, essendo l'uscita di un flip-flop D il suo stato all'accensione della scheda sarà indefinito.
Nel caso in cui all'accensione il livello di ELCD sia alto lo si resetta eseguendo una lettura a vuoto sul display, infatti ad ogni lettura si commuta lo stato del flip-flop, questa operazione la si eseguirà solo nella fase di inizializzazione del software, dato che ogni volta che si vorrà scrivere o leggere un dato nel display, si effettuerà una doppia lettura o scrittura riportando così lo stato di ELCD sempre a 0. La routine che esegue questo controllo si chiama INILCD e si trova nel modulo PERI.ASM.
Come anticipato si accede al display tramite due registri, INSTREG lo si può vedere come sia un registro di controllo che di stato, mentre DATAREG è un registro dati, in pratica si utilizza INSTREG per indicare a che registro interno del display si vuole accedere, dopodiché si trasferiscono i dati in tale registro scrivendoli in DATAREG, è cosi possibile impostare tutti i parametri del display. Il registro INSTREG permette inoltre di capire se il display è pronto a ricevere o cedere dati, tramite il flag Busy.
In conclusione per scrivere dati nel display in un determinato registro, occorrono due scritture all'indirizzo INSTREG, e poi due scritture con il valore che vogliamo impostare all'indirizzo DATAREG, per poter riaccedere ulteriormente sul display occorre vedere lo stato del flag Busy

ed attendere finché non va a livello 0, la routine che testa tale flag si chiama TSTBUSY e si trova nel modulo PERI.ASM, è qui di seguito riportata.

```
TSTBUSY EQU *
      PSHB
TSTB_0:
      LDAB    INSTREG
      LDAB    INSTREG
      ANDB    #BUSYFLG
      BNE     TSTB_0
      PULB
      RTS
```

Ecco un esempio di settaggio impostazioni del display:

```
      LDAB    #$01            ; ---Settaggio area carattere
      STAB    INSTREG         ; =
      STAB    INSTREG         ; =
      LDAB    #$77            ; Vp=8 Hp=8
      STAB    DATAREG         ; =
      STAB    DATAREG         ; =
      JSR     TSTBUSY         ; Attesa
      LDAB    #$02            ; ---Set numero di caratteri
      STAB    INSTREG         ; =
      STAB    INSTREG         ; =
      LDAB    #$1D            ; Hn=30 caratteri per riga
      STAB    DATAREG         ; =
      STAB    DATAREG         ; =
```

In questo caso scrivendo il valore $01 nel registro INSTREG si accede al registro del display in cui si imposta l'area del carattere, dopo di che scrivendo nel DATAREG il valore $77 si imposta l'area del carattere pari a 8 pixel di altezza per 8 di larghezza, per poter poi accedere al registro di impostazione numero caratteri occorre prima lanciare la routine TSTBUSY ed attendere finché il display non è pronto.

Tutte le impostazioni iniziali del display da settare all'accensione o al reset del sistema si trovano nella routine INIDISP nel modulo PERI.ASM.

4.3.3 Driver di comunicazione seriale SCI (Serial Communication Interface)

Il formato dei dati seriali è costituito dalle seguenti parti:

1) Un bit di start di livello logico zero, sia in trasmissione che in ricezione

2) 8 bit di dati, spediti a partire da quello meno significativo

3) Un bit di stop di livello logico alto

L'interfaccia seriale del 68HC11 è costituita da vari registri, un registro dati SCDR, tre registri di controllo SCCR1, SCCR2 e BAUD, e un registro di stato SCSR, per effettuare la trasmissione e la ricezione seriale l'interfaccia utilizza uno Shift Register che legge o scrive i dati nel registro

SCDR, in pratica quest'ultimo registro assolve la funzione di Buffer, infatti grazie a questi è possibile precaricare il registro dati SCDR mentre lo shift register sta ancora shiftando (inviando i dati).

Il registro di controllo SCCR1 si occupa di impostare il formato dei dati a 8 o 9 bit, e di memorizzare il nono bit sia trasmesso che ricevuto, in questo applicazione non abbiamo previsto l'utilizzo di dati a 9 bit per cui questo registro non verrà mai impostato.

Il registro SCCR2 contiene vari bits di controllo, quelli utilizzati in questa applicazione sono:

TIE - (Trasmit Interrupt Enable) settando questo bit, nel caso in cui il registro dati sia vuoto, si genera la richiesta di interrupt SCI

RIE - (Receiver Interrupt Enable) settando questo bit, nel caso in cui il registro dati sia pieno, si genera la richiesta di interrupt SCI

TE - (Trasmit Enable) quando commuta da 0 a 1 oltre ad attivare la trasmissione spedisce una trama di bit tutti a 1

RE - (Receiver Enable) settato abilita la ricezione.

Il registro di SCSR contiene vari flags di stato quelli utilizzati sono:

TDRE - (Trasmit Data Register Empty Flag) quando il registro dati è vuoto questo flag si setta, per resettarlo occorre prima leggerlo e dopo eseguire la scrittura in SCDR

RDRF - (Receive Data Register Full Flag) quando il registro dati è pieno questo flag si setta, e per resettarlo occorre prima leggerlo e dopo eseguire una lettura di SCDR

Infine il registro di controllo BAUD come dice il nome stesso serve ad impostare il baud rate lo stesso sia per la trasmissione che per la ricezione, contiene 3 bit (prescaler bit) per impostare il massimo baud rate ed altri 3 bit (select bits) per selezionare un ulteriore sottomultiplo, il limite minimo di baud rate vale 75 byte/sec il massimo vale 125000 byte/sec con una frequenza di clock di 8 Mhz.

Per velocizzare l'esecuzione del programma si è preferito accedere all'interfaccia seriale sia in trasmissione che in ricezione tramite interrupt, ossia ogni volta che si vuole spedire un carattere su seriale si abiliterà l'interrupt di trasmissione, mentre in ricezione l'abilitazione di interrupt sarà sempre attiva, poiché nel 68HC11 la richiesta di interrupt della seriale SCI è una sola, sia per la trasmissione che per la ricezione la routine di interrupt lanciata sarà una sola, che dovrà distinguere così i due casi.

Prima di analizzare la routine di interrupt della seriale è bene dare una visione generale di come l'intero programma accede sia in lettura che in scrittura ai dati della seriale, infatti come si

osserva da figura 4-5 tutti i dati sia ricevuti che quelli che si vuol trasmettere passano attraverso due buffer circolari che permettono così di accumulare caratteri evitandone la perdita.

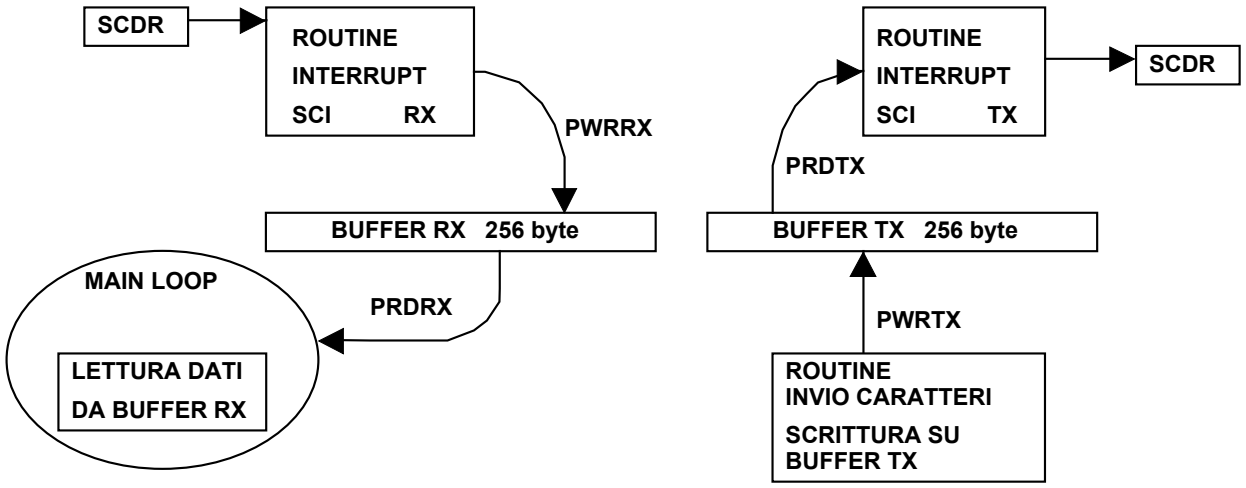


Figura 4-E

Il motivi per cui questi buffer sono circolari è dato dal fatto che, come si osserva da entrambi gli schemi a blocchi di figura 4-5, tutti i caratteri vengono letti o scritti in trasmissione e ricezione nei relativi Buffer RX e TX, nella posizione indicata dai quattro puntatori PWRRX, PRDRX, PWRTX, PRDTX, dopodiché vengono incrementati i valori dei puntatori per le successive operazioni, quando hanno raggiunto il valore di 255 tornano a 0 riniziando così a scrivere o leggere all'inizio dei buffers relativi.

In pratica ogni volta per poter scrivere in un buffer occorre controllare che il puntatore di scrittura incrementato non sia uguale a quello di lettura, altrimenti si avrebbe sovrapposizione di dati, per quanto riguarda la lettura dovrà essere eseguita ogni volta che i puntatori (RD e WR) sono diversi fra di loro, infatti in questo caso vuol dire che ci sono caratteri nuovi nel buffer, chiaramente una volta letto il carattere andrà incrementato il puntatore di lettura. Queste considerazioni valgono sia per il buffer RX che per il TX.

In caso di ricezione la sezione che si occupa di vedere se ci sono caratteri da leggere, e nel caso affermativo di acquisirli, si trova all'interno del main loop, infatti anche nel caso in cui ci siano molti caratteri nuovi nel buffer RX se ne leggerà solo uno ad ogni ciclo di main, questo permette di non far stazionare il programma troppo tempo nel solito punto, ma di processare così un carattere alla volta e nel caso in cui si tratti di sequenze di caratteri tener conto della loro posizione temporale come si vedrà più avanti.

Nel caso in cui si voglia spedire un carattere su seriale occorrerà prima scriverlo nel buffer TX controllando che non sia pieno e allo stesso tempo abilitare l'interrupt SCI di trasmissione,

dopodiché partirà automaticamente la routine di interrupt SCI che eseguirà le operazioni relative alla trasmissione. La routine che si occupa di scrivere il carattere nel buffer TX si chiama SENDCHR e si trova nel modulo SUBG.ASM.
La routine di interrupt che si occupa di scrivere o leggere i caratteri nel registro dati SCDR si chiama SCIRXTX, si trova nel modulo INT.ASM è qui di seguito riportato il codice:

```
*
*-----------------------Routine di Interrupt SCI--------------------------
*
SCIRXTX EQU     *
        LDAB    SCSR    ; Lettura registro di stato seriale
        ANDB    #TDRE   ; Controlla se il registro dati e' vuoto
        ANDB    SCCR2   ; e se e' abilitato l'interrupt di trasmissione
        BNE     SCITX   ; se nessuna delle condizioni e'valida allora
*                       ; si tratta di Ricezione
*                       -------RICEZIONE--------
        LDAA    SCDR            ; Lettura carattere da registro dati
        LDAB    PWRRX           ; Lettura puntatore scrittura
        INCB                    ; Controlla se il buffer RX e' pieno
        CMPB    PRDRX           ; se e' pieno uscire
        BEQ     SCI_R           ; =
        STAB    PWRRX           ; Altrimenti aggiorna puntatore scrittura
        DECB
        LDX     #RXBUF          ; Carica buffer RX
        ABX
        STAA    0,X             ; Scrittura carattere nel BufferRX
        BRA     SCI_R
*                       ------TRASMISSIONE------
SCITX:
        LDAB    PRDTX           ; Lettura puntatore lettura
        CMPB    PWRTX           ; Confronto col puntatore scrittura
        BEQ     SCIT_R          ; esci se il bufferTX non ha caratteri nuovi
        LDX     #TXBUF          ; Carica Buffer TX
        ABX
        INCB                    ; Incremento puntatore readTX
        STAB    PRDTX
        LDAB    0,X             ; Lettura carattere da BufferTX
        STAB    SCDR            ; Scrittura carattere nella seriale
        BRA     SCI_R
SCIT_R:
        BCLR    SCCR2,#TIE      ; Disabilitazione interrupt di TX
SCI_R:
        RTI
```

Come si osserva in testa alla routine viene fatto il controllo per capire se l'interrupt che si è scatenato è di ricezione o trasmissione, in base al registro di stato ed al registro di controllo, dopodiché in caso di ricezione si controlla se c'è posto nel buffer RX e nel caso affermativo ci si può scrivere il carattere letto da SCDR; in caso di trasmissione si controlla se i puntatori sono diversi fra loro, ossia se ci sono caratteri nuovi nel buffer TX, e nel caso affermativo si procederà a copiare il carattere dal buffer al registro SCDR, ottenendo così la spedizione su seriale, è da notare che la richiesta di interrupt di trasmissione rimane attiva finché non sono stati spediti tutti

i caratteri, infatti solo in tal caso si resetta il bit di controllo TIE nel registro SCCR2, come si nota nell'ultima riga del codice.

4.3.4 Driver interfaccia tastiera:

Il driver della tastiera ha la funzione di scandire ad ogni ciclo di main ogni riga della tastiera e di leggere le colonne relative, per ogni colonna attiva occorre poi calcolare il codice di scansione del tasto relativo, il driver deve però rispettare varie specifiche, come anticipato deve essere in grado di leggere la pressione contemporanea di più tasti, e deve inoltre effettuare un sistema di antirimbalzo software.

Per quanto riguarda il sistema di antirimbalzo la tecnica utilizzata è quella di considerare validi solo i tasti che risultano attivi per due volte consecutive ossia per due chiamate della routine driver, infatti in tal caso si possono considerare esauriti gli impulsi di rimbalzo, per fare questo si fa uso di due buffer in un primo buffer si salva ogni volta il codice di scansione relativo al tasto premuto, se però il codice è già presente nel primo buffer allora lo si può copiare nel secondo, questo secondo buffer dovrà essere in grado di contenere fino a quattro codici scansione ossia quattro tasti premuti contemporaneamente.

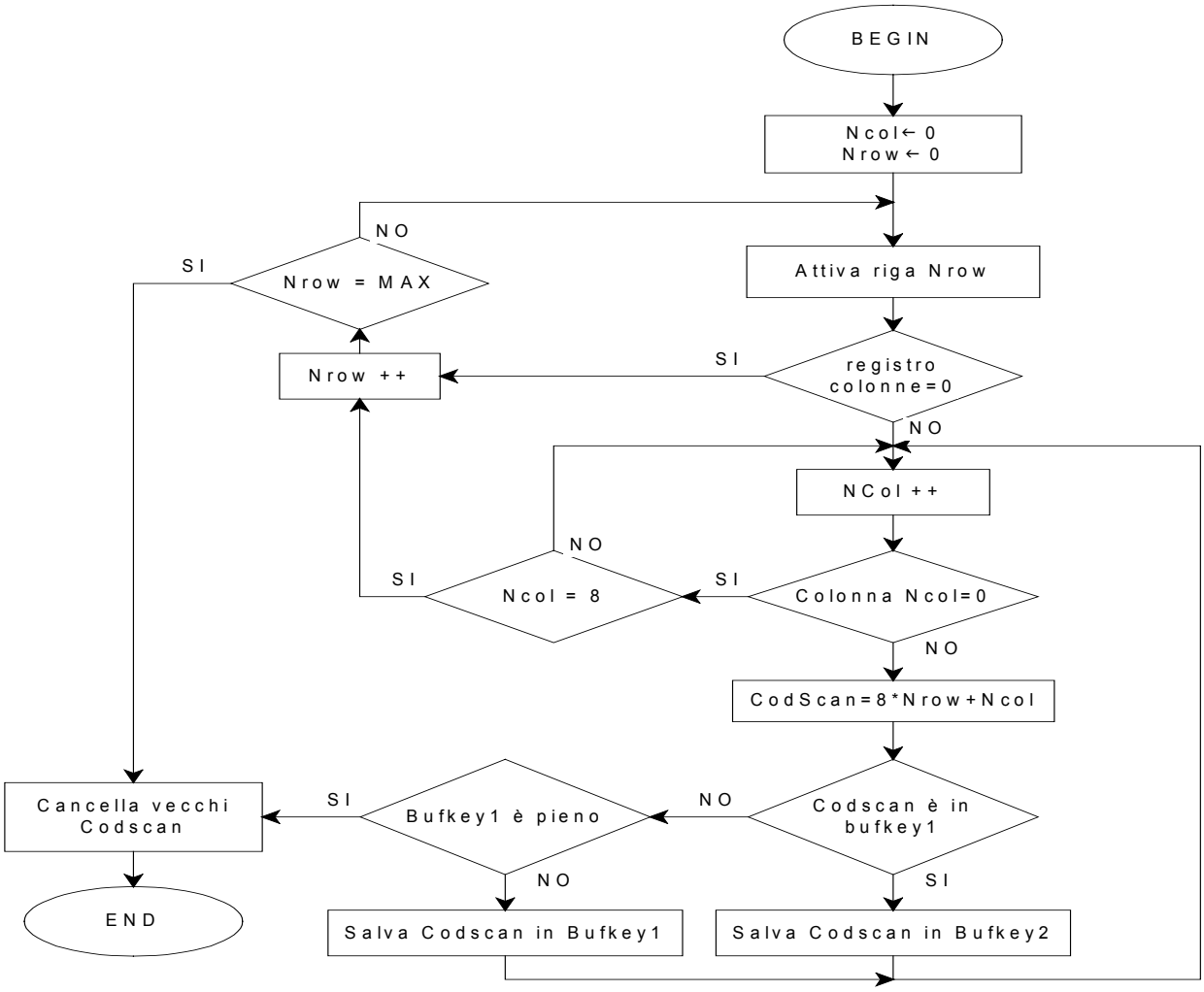


Figura 4-F

Come si nota dallo schema a blocchi di figura 4-6, la scansione viene effettuata attivando una riga alla volta, se nessuna colonna è attiva si va ad attivare la riga successiva, nel caso in cui il valore del registro colonne è diverso da zero allora si testano tutti i bit è per ognuno attivo si va a calcolare il codice di scansione del tasto.

Una volta calcolato il codice di scansione lo si va a ricercare nel bufkey1 se è già presente allora lo si può salvare in bufkey2, altrimenti lo si mette in bufkey1, questa operazione viene ripetuta per tutti i tasti premuti, una volta scandita tutta la tastiera si cancellano tutti gli scancode presenti in bufkey1 escluso gli ultimi arrivati, che serviranno per la successiva chiamata di questa subroutine.

Terminata la routine Bufkey2 sarà vuoto o conterrà i codici scansione dei tasti premuti contemporaneamente fino ad un massimo di 4.

4.4 Fase di inizializzazione

Sono di seguito descritte le operazioni che vengono eseguite ogni volta che viene avviato il terminale, oppure ogni volta che si genera uno reset software.

4.4.1 Modo di funzionamento e mappatura memoria del microcontrollore:

Il microcontrollore 68HC11, possiede quattro modi di funzionamento due normali e due di test, selezionabili durante il reset tramite i pin MODA e MODB.

Come anticipato in questa scheda è stato utilizzato il modo Expanded, che permette di utilizzare oltre alla memoria interna del micro anche memoria esterna.

È inoltre possibile tramite un apposito registro del micro indicare dove posizionare i 256 byte di RAM interna e il blocco da 64 byte di registri del micro all'interno dello spazio memoria. Entrambi gli spazi possono essere traslati a passi di 4 Kbyte, per spezzettare il meno possibile la memoria come si osserva da figura 4-7 si è preferito posizionare entrambi gli spazi a partire dall'indirizzo $0000, l'unico svantaggio è che si perdono 64 byte di RAM interna, dato che i registri hanno una priorità maggiore rispetto ad essa, riducendola così a 192 byte, aspetto irrilevante data la grande capacità della RAM esterna. La RAM interna ha come vantaggio rispetto a quella esterna di potervi accedere con un indirizzamento diretto, ossia con un indirizzo di solo un byte, anziché due necessari per quella esterna (Indirizzamento esteso).

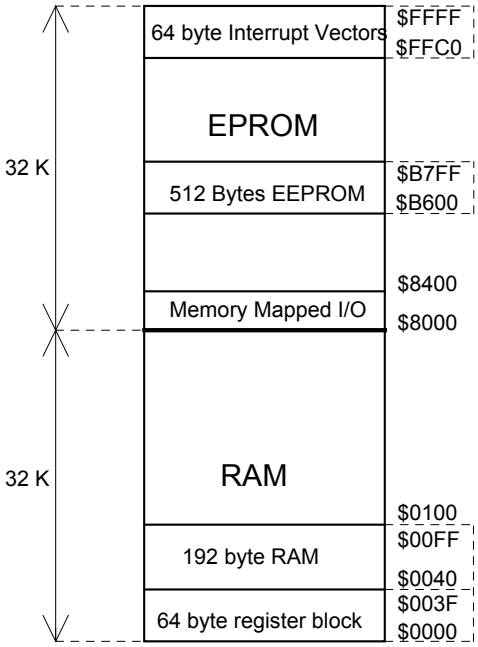


Figura 4-G

Dalla mappatura della memoria si osserva inoltre uno spazio di 64 byte riservato ai vettori di interrupt a partire dall'indirizzo $FFC0 fino alla fine dello spazio totale di indirizzamento.

A partire dall'indirizzo B600 sono disponibili inoltre 512 Bytes di EEPROM, utili per salvare ad esempio delle variabili del programma che si desidera mantenere anche dopo lo spegnimento.

4.4.2 Inizializzazione variabili:

Al momento dell'accensione o del reset alcune variabili dovranno essere inizializzate con dei valori predefiniti, il terminale però prevede che alcune impostazioni, modificabili durante la fase di Setup, debbano essere mantenute anche dopo lo spegnimento, per cui per queste si dovrà far uso della memoria EEPROM. In pratica ad ogni accensione si dovranno leggere le impostazioni salvate precedentemente in EEPROM e trasferirle nelle variabili in RAM, per risolvere il problema della prima accensione in assoluto, si è inserito un controllo della EEPROM che permette di vedere se i dati sono errati o se non sono mai stati salvati, così nel caso di fallimento del test si provvede a inizializzare tali variabili con dei valori di base.

La tecnica di controllo utilizzata è quella del Checksum, con la quale si controlla che la somma dei valori di tutti i bytes salvati sia uguale al valore scritto negli ultimi 2 bytes, ovviamente ogni volta che si copieranno le variabili in EEPROM si dovrà calcolare la loro somma e salvarla negli ultimi 2 bytes; la routine che esegue questo controllo si chiama CHCKSUM e si trova nel modulo SUBG.ASM.

Nel caso di fallimento del test Checksum, oltre ad impostare le variabili di SETUP con dei valori di base si ripete il test Checksum, nel caso in cui continui a fallire per più di due volte allora si emette un messaggio di errore e si blocca il programma, questo situazione si verificherà solo nel caso in cui la memoria EEPROM, contenuta all'interno del micro, si sia danneggiata.

4.4.3 Routines di inizializzazione:

In questa ultima parte della fase di inizializzazione vengono lanciate le routine di inizializzazione delle periferiche viste in precedenza, e viene inoltre pulito completamente lo schermo del display, a questo punto è possibile entrare nel Main Loop rendendo così il terminale pronto per la ricezione e la trasmissione.

4.5 Main Loop

Passerò adesso ad analizzare le varie parti che compongono il main loop:

4.5.1 Aggiornamento variabile duty-cicle PWM:

La variabile THPWM utilizzata dalla routine di interrupt PWM viene aggiornata ad ogni ciclo di main loop in base al peso sia del potenziometro che del sensore di temperatura NTC, inoltre nel caso in cui il potenziometro sia assente la regolazione manuale potrà essere effettuata da tastiera in fase di Setup.

La funzione che fornisce la variabile THPWM sarà data dalla seguente combinazione lineare $THPWM = m_1 \cdot V_{POT} + m_2 \cdot V_{NTC} + c$, dove V_{POT} è il valore digitale della tensione del potenziometro o della variabile modificabile da Setup, V_{NTC} è il valore digitale della tensione fornita dall'NTC.

Nella tabella sottostante sono riportati i valori che deve avere THPWM in funzione della temperatura, sono riportati inoltre i valori analogici della tensione Vntc che fornisce il sensore di temperatura e i relativi valori convertiti in digitale.

T(°C)	Vlcd	Duty Cicle %	THPWM (digit.)	Vntc (V)	Vntc (digit.)
0	11.9	79.3	202	2.234	114
25	10.8	72	184	1.057	51
40	10.4	69.3	177	0.645	33

Come si osserva da tabella ad una variazione di 40°C deve corrispondere una variazione di THPWM di 202-177=25 per cui per il calcolo del coefficiente m2 occorre imporre che $m2 \cdot (114 - 33) = 25$, si ricava così m2 pari a 25/81=0.31.

Per ricavare il valore dei coefficienti m1 e c si è imposto solamente che alla temperatura di 25°C con il potenziometro a metà corsa il valore di THPWM valga 184, ossia si è imposto che $m1 \cdot Vpot + c = 184 - 51 \cdot m2 = 168$, dopodiché in base a delle prove manuali si è ottenuto una giusta sensibilità ed escursione di contrasto con i valori di m1=0.3 e c=128; in conclusione la funzione che si ottiene è: .

Come anticipato in assenza del potenziometro al posto della variabile Vpot si dovrà sostituire la variabile modificabile da tastiera in fase di Setup chiamata VBRIGHT, senza nessuna modifica dei coefficienti della funzione, infatti anche per questa variabile l'escursione va da 0 a 255 come per il potenziometro.

4.5.2 Gestione tastiera e trasmissione su seriale:

Secondo lo standard VT100 e VT52 si può suddividere la tastiera in 3 gruppi, tasti standard, tasti di controllo, e tasti della tastierina numerica, a questi tipi di tasti corrisponderanno relativi caratteri da trasmettere su seriale.

Per tasti standard si intendono tutti i tasti alfanumerici classici ai quali corrispondono i caratteri secondo lo standard ASCII, tutti questi tasti sono sensibili alla pressione dello Shift, mentre sono sensibili anche allo stato del Caps-Lock solo quelli alfabetici.

Per tasti di controllo si intendono tutti quei tasti ai quali non corrisponde un carattere visualizzabile, ma bensì un codice di controllo al quale è associata una determinata azione, ad esempio il tasto ENTER. I caratteri di controllo possono essere generati direttamente come nel

caso dell'Invio, oppure tramite la pressione contemporanea di alcuni tasti con il tasto Ctrl, per la lista dei tasti di controllo vedere nel capitolo caratteri trasmessi nell'appendice.
Per il gruppo di tasti della tastierina numerica lo standard di trasmissione VT100-VT52 ha previsto fino a 4 funzioni diverse per ognuno di questi tasti.
In base allo stato del NUM-LOCK, i tasti della tastierina numerica possono assolvere la funzione di tasti cursore o di tasti numerici, i caratteri che generano dipendono inoltre dalla modalità in cui sono stati impostati dall'esterno (dal computer), per i tasti cursore si può avere la modalità Set o Reset, mentre per i tasti numerici si può avere la modalità Application o Normal.
I tasti cursore Set o Reset ed i tasti numerici in Application mode devono a differenza degli altri generare non un solo carattere ma una sequenza di due caratteri per lo standard VT52 e di tre per lo standard VT100, ad esempio al tasto Cursor Up il VT52 associa la sequenza "ESC A", mentre il VT100 "ESC [A", lo standard VT 52 si differenzia dal VT100 inoltre per possedere una sola modalità per i tasti cursore; tutte le sequenze che devono essere generate da i due standard sono riportate in appendice.

4.5.2.1 Conversione codici di scansione:

Come anticipato la routine di scansione SCANKEY rende nel buffer BUFKEY2 fino a 4 codici scansione dei tasti premuti, occorre però un'ulteriore elaborazione che permetta di associare al codice di scansione il giusto carattere ASCII, oppure la giusta sequenza da trasmettere, la routine che si occupa di convertire il codice di scansione si chiama CONVKEY e si trova nel modulo SUBG.ASM.
La tecnica di base per effettuare la conversione dei tasti è quella di utilizzare una tabella contenente i codici ASCII di tutti i tasti e di sfruttare il valore del codice di scansione del tasto come offset all'interno della tabella per prelevare il codice ASCII corrispondente; in realtà sono necessarie più di una tabella dato che ad uno stesso tasto possono corrispondere più di un carattere ASCII, ad esempio nel caso dei tasti alfabetici si possono avere o maiuscole o minuscole, inoltre ad alcuni tasti non va associato un codice ASCII, ma bensì una sequenza.
Il primo problema da risolvere per cui è quello della scelta delle varie tabelle di conversione, la scelta dovrà essere effettuata in base allo stato dei tasti funzione come Shift, Ctrl, Alt, ma anche in base alla sensibilità o meno del tasto ad essi. La tecnica usata per cui è stata quella di utilizzare una prima tabella dove sono riportate le proprietà di ogni tasto, dopodiché in base alla proprietà del tasto letta ed alla pressione dei tasti funzione si potrà scegliere la giusta tabella di conversione.

Per risolvere il problema delle sequenze la soluzione è stata di inserirle in una tabella a parte; per i tasti che devono generare sequenze i corrispondenti codici nelle tabelle di conversione assumono un significato diverso, infatti diventano puntatori alla tabella sequenze, per distinguere carattere da puntatore si è sfruttato il fatto che i singoli caratteri ASCII da trasmettere hanno un valore compreso fra 0 e 127 decimale, per cui si è utilizzato i valori superiori a 127 come puntatori per la sequenza da selezionare successivamente nella tabella delle sequenze. In conclusione le tabelle di conversione conterranno al loro interno sia codici ASCII che puntatori.
La routine di conversione CONVKEY riassumendo dovrà eseguire le seguenti operazioni:
1) Leggere tutti i codici di scansione contenuti in BUFKEY2.
2) Verificare se sono presenti i codici scansione di Shift, Ctrl, Alt, Caps-Lock e Num-Lock, ed in caso affermativo attivare i relativi flags contenuti in un apposito byte KEYFLG.
3) Leggere le proprietà del tasto premuto e selezionare in base ai flags contenuti in KEYFLG la giusta tabella di conversione.
4) Leggere dalla tabella di conversione selezionata il codice ASCII o il puntatore alla sequenza da trasmettere.

4.5.2.2 Gestione autorepeat:

Lo standard VT100-VT52 prevede l'autorepeat, ossia la ripetizione del carattere in caso di pressione prolungata di un tasto della tastiera, come in un normale PC.
Per gestire l'autorepeat occorre tener conto del tempo di ritardo prima che inizi la ripetizione e della velocità di ripetizione, dato che la tastiera viene letta ad ogni ciclo di main se si trasmettesse ad ogni ciclo lo stesso carattere la velocità sarebbe troppo elevata per cui come prima cosa si è pensato di far eseguire la routine di conversione dei tasti (descritta sopra) ogni 10 msec anziché ad ogni ciclo.
Con questo primo ritardo si impone la massima velocità di autorepeat, per ridurla ulteriormente si è ricorso ad una variabile contatore che viene decrementata ogni volta che si esegue CONVKEY, solo al suo azzeramento si procederà con la trasmissione del carattere ASCII o della sequenza.
In base al valore con cui viene inizializzato il contatore si regolerà la velocità di autorepeat, lo stesso contatore viene utilizzato anche per generare il ritardo prima che inizi la ripetizione, entrambi questi parametri sono impostabili in fase di setup.

L’intero blocco che si occupa di gestire l’autorepeat si trova direttamente nel modulo MAIN.ASM è posto fra la routine CONVKEY e la routine che si occupa di trasmettere i caratteri o le sequenze.

4.5.2.3 Trasmissione carattere o sequenza:

Questa routine si occupa di distinguere se il valore reso da CONVKEY è un carattere ASCII oppure un puntatore e procederà così ad inviare su seriale o direttamente il codice ASCII oppure a andrà a prelevare dalla tabella delle sequenze la giusta sequenza di trasmissione.
In realtà le tabelle delle sequenze sono due, una per lo standard VT100 e l’altra per il VT52, la routine di trasmissione deve per cui scegliere quale delle due tabelle utilizzare in base allo standard richiesto, la selezione dello standard può essere fatta o in fase di Setup oppure dall’esterno tramite computer.
Nel diagramma di flusso sottostante è rappresentata l’intera gestione della tastiera e della trasmissione a partire dalla routine di scansione SCANKEY fino ad arrivare alla routine di trasmissione SENDKEY, tutto il blocco rappresentato si trova all’interno dell’anello Main loop e come si osserva mentre SCANKEY viene eseguita ad ogni ciclo di main il resto viene eseguito ogni 10 msec.

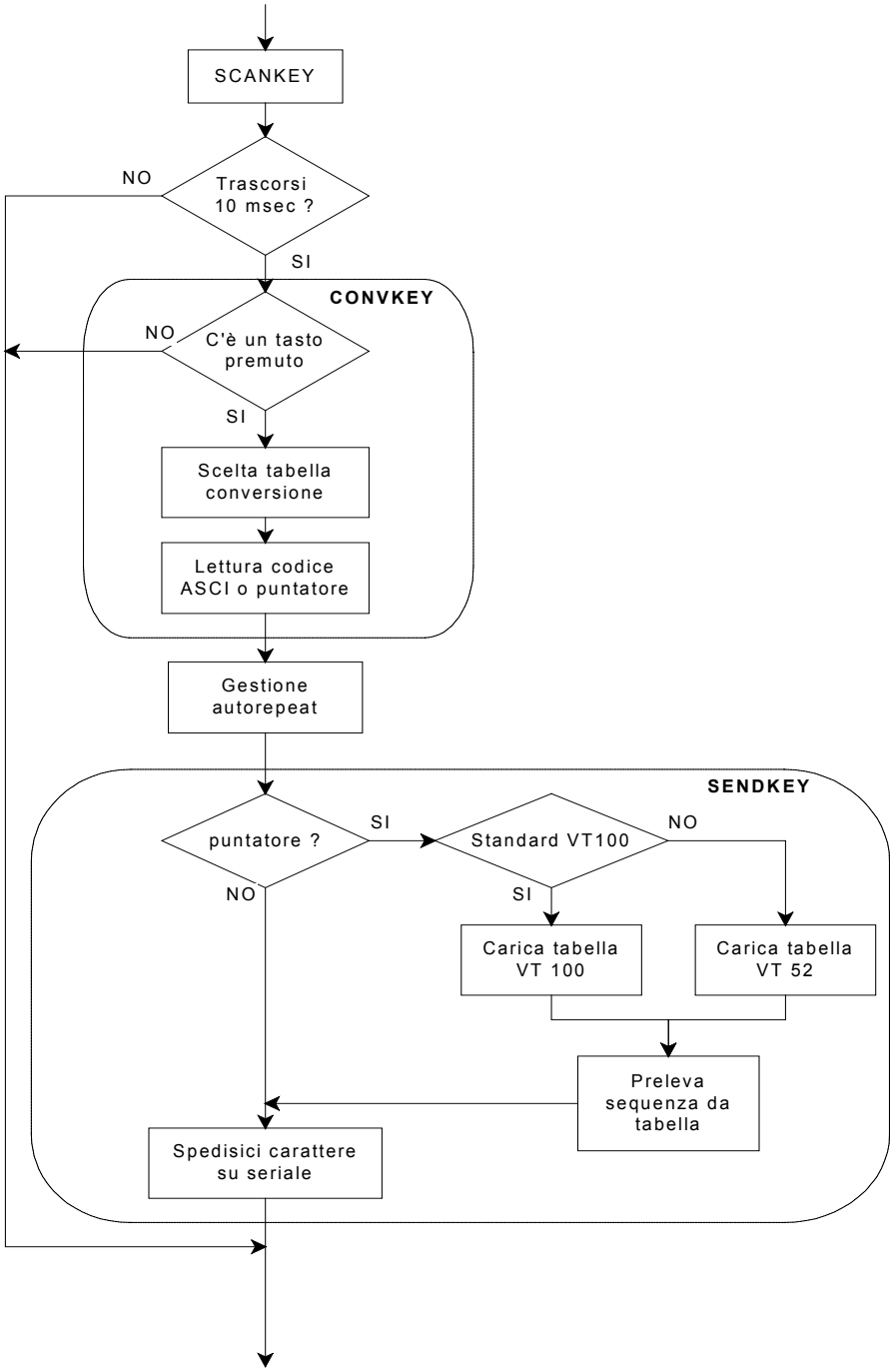


Figura 4-H

4.5.3 Ricezione e analisi dei caratteri:

Per la ricezione come per la trasmissione si ha una suddivisione fra caratteri singoli e sequenze di caratteri, i caratteri singoli si dividono a sua volta in caratteri visualizzabili e caratteri di controllo.

I caratteri di controllo insieme alle sequenze non devono essere stampati sullo schermo, ma devono eseguire un azione specifica sul terminale, ad esempio modificare delle impostazioni

oppure spostare il cursore, eseguire dei test ecc. i caratteri di controllo riconosciuti dal terminale sono descritti in appendice nella tabella **Caratteri di controllo generati**.

Le sequenze si suddividono secondo lo standard ANSI in due categorie, sequenze di escape e sequenze di controllo, lo standard VT100 prevede l'utilizzo di entrambe a differenza dello standard VT52 che riconosce solo le sequenze di escape.

Il formato di una sequenza di escape è il seguente:

ESC I...I F

il carattere 'ESC' è l'introduttore di sequenza, ogni volta che si riceve questo carattere i successivi non devono essere stampati sullo schermo.

Dopo l'introduttore ESC (27 dec.) si possono avere zero o più di un carattere intermedio "I" il cui valore ASCII è compreso fra 32 e 47 decimale; la sequenza terminerà non appena sarà ricevuto il carattere finale "F" il cui valore è compreso fra 48 e 126, a questo punto il terminale deve eseguire l'azione relativa e tornare a stampare i successivi caratteri printabili .

Per quanto riguarda lo standard VT100 tutte le sequenze di escape riconosciute hanno al massimo un solo carattere terminale, un esempio è il seguente:

ESC (B

Per quanto riguarda le sequenze di controllo la loro forma generica utilizzata dallo standard VT100 è la seguente:

ESC [Ps ; ..... ; Ps F

In pratica ESC [è l'introduttore di sequenza, mentre Ps sono dei parametri che possono essere caratteri singoli o stringhe, il ';' funge da carattere separatore fra i vari parametri; 'F' è il carattere terminale, che in questo caso ha la funzione di selezionare il tipo di azione da eseguire.

Un esempio di sequenza di controllo è la seguente:

ESC [2 ; 1 0 r

in questo caso i parametri rappresentano variabili numeriche, le singole cifre sono trasmesse sotto forma di caratteri ASCII, questa sequenza imposta i margini superiori e inferiori della regione di scrolling rispettivamente alla riga numero 2 e alla riga numero 10.

Se durante una sequenza vengono ricevuti i caratteri CANcel (24 dec.) o SUBstitute (26 dec.) la sequenza viene interrotta ed i successivi caratteri potranno essere stampati sullo schermo.

Per quanto riguarda le situazioni di errore lo standard non prevede nessuna azione specifica, ad esempio nel caso che si ricevano caratteri di controllo non riconosciuti il terminale dovrà ignorarli, e così nel caso di parametri non corretti all'interno di una sequenza il terminale oltre a

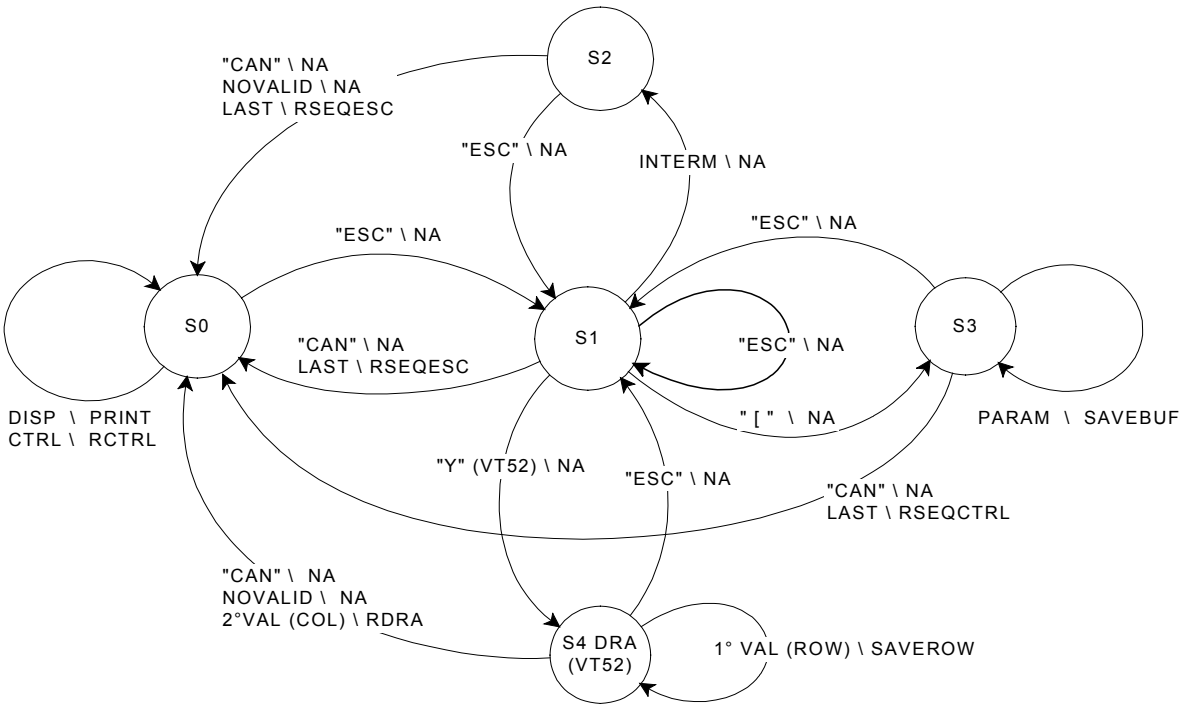
non dover eseguire nessuna azione dovrà permettere appena riconosciuto l'errore che i successivi caratteri possano essere stampati sullo schermo.

4.5.3.1 Gestione ricezione:

Lo standard ANSI non prevede nessun legame temporale fra un carattere e l'altro delle sequenze, i caratteri possono arrivare con qualsiasi ritardo, dunque una volta ricevuto il carattere di Start (ESC) della sequenza non si può attendere l'arrivo di tutti i restanti caratteri, ma occorre uscire dalla fase di ricezione ed eseguire il resto delle operazioni del ciclo di main.

La tecnica utilizzata per gestire l'intera ricezione è quella della macchina a stati, ossia suddividere la fase di ricezione in più stati e in base al tipo di carattere ricevuto passare da uno stato all'altro ed eseguire l'azione relativa, in pratica ogni carattere ricevuto sarà processato in maniera diversa a seconda in che stato si trovi la ricezione. Gli stati in realtà non saranno altro che delle routine che verranno lanciate dal Main loop ogni volta che si riceve un carattere, per selezionare la giusta routine si utilizza una variabile di stato.

Gli stati necessari per gestire la ricezione di tutti i tipi di caratteri e sequenze sia per il VT100 che per il VT52 sono cinque, nella figura 4-9 è riportato il diagramma degli stati completo della ricezione.



TIPO DI CARATTERE RICEVUTO \ AZIONE DA ESEGUIRE

TIPI DI CARATTERI RICEVUTI:

DISP = CARATTERE PRINTABILE
CTRL = CARATTERE DI CONTROLLO
INTERM = CARATTERE INTERMEDIO SEQUENZA
PARAM = CARATTERE PARAMETRO SEQUENZA
LAST = CARATTERE TERMINALE SEQUENZA
NOVALID = CARATTERE ERRATO SEQUENZA
1°VAL (ROW) = 1° VALORE SEQUENZA DRA (RIGA)
2°VAL (COL) = 2° VALORE SEQUENZA DRA (COLONNA)

AZIONI DA ESEGUIRE:

PRINT = STAMPA CARATTERE SU DISPLAY
RCTRL = ESEGUI ROTUINE CARATTERE DI CONTROLLO
NA = NESSUNA AZIONE
RSEQESC = ESEGUI ROUTINE SEQUENZA DI ESCAPE
RSEQCTRL = ESEGUI ROUTINE SEQUENZA DI CONTROLLO
SAVEBUF = SALVA PARAMETRI NEL BUFFER
SAVEROW = SALVA 1° VALORE SEQUENZA DRA (ROW)
RDRA = ESEGUI ROUTINE DRA (DIRECT CURSOR ADDRESS)

Figura 4-I

4.5.3.2 Descrizione delle routines di stato:

Lo stato di partenza è S0, in questo stato viene confrontato il carattere ricevuto con la lista dei caratteri di controllo, nel caso in cui vi appartenga verrà eseguita la relativa routine di controllo, altrimenti significa che il carattere è printabile per cui verrà stampato sul display alla posizione del cursore.

Nel caso in cui il carattere ricevuto sia ‘ESC’ allora verrà aggiornata la variabile di stato da 0 ad 1, così che il prossimo carattere verrà elaborato dallo stato S1.

Per quanto riguarda la stampa del carattere, in questo stato viene effettuato anche lo scrolling nel caso in cui il cursore abbia raggiunto la fine dello schermo.

Lo stato S1 si occupa di analizzare il carattere successivo ad ‘ESC’, di capire di che tipo di sequenza si tratti e di conseguenza aggiornare la variabile di stato futuro. Come si vede dal

diagramma di flusso da questo stato si potrà passare agli stati S2, S3, S4 (DRA), oppure si potrà tornare indietro allo stato S0.

Nello stato S1 inoltre si possono subito riconoscere le sequenze di escape che non hanno caratteri intermedi ed eseguire così la relativa routine, in questo caso lo stato futuro diventerà S0.

Lo stato S2 si occupa di riconoscere le sequenze di escape con un carattere intermedio, in pratica verifica se il carattere ricevuto è un carattere terminale, nel caso in cui lo sia verrà eseguita la relativa routine, altrimenti in caso di carattere non valido si tornerà allo stato S0.

Lo stato S3 si occupa delle sequenze di controllo, ossia delle sequenze con parametri, una volta entrati in questo stato tutti i caratteri ricevuti vengono salvati in un buffer, finché non giunge un carattere terminale, in questo caso verrà lanciata la routine relativa in base al solo carattere terminale, sarà poi la routine stessa a controllare i parametri leggendoli dal buffer. Nel caso in cui non giunga mai il carattere terminale per evitare che non si esca più da questo stato, si sono inseriti dei controlli sul numero massimo di parametri e sul numero massimo di caratteri per parametro, se non si rispettano queste due condizioni si considera la sequenza errata e si torna allo stato S0.

Lo stato S4 (DRA) è uno stato particolare, infatti viene utilizzato solo in caso di emulazione VT52 e serve per riconoscere una sola sequenza denominata DRA (Direct Cursor Address), questa è una sequenza particolare dato che non possiede carattere terminale ma ha però una lunghezza fissa come si vede dalla sua forma generica:

ESC Y row+32 col+32

in pratica si rimane in questo stato finché non si sono ricevuti entrambi i parametri, dopodiché la routine di stato stessa effettuerà il posizionamento del cursore, ricavando i valori di riga e di colonna dai parametri.

È da notare dal diagramma a stati che in qualunque stato si trovi il sistema si tornerà nello stato S0 ogni volta che si riceve un carattere errato ossia non appartenente a nessuna sequenza, oppure nel caso che si riceva il carattere ‘CAN’ che rappresenta l’interruttore di qualsiasi sequenza in entrambi questi casi non si dovrà eseguire nessuna azione a parte impostare la variabile di stato a 0. Si osserva inoltre che il carattere introduttore ‘ESC’ in qualsiasi stato sia ricevuto farà andare il sistema nello stato S1.

5. UTILIZZO DEL TERMINALE LCD

5.1 Modalità SET-UP

Tutte le impostazioni del terminale possono essere modificate entrando nella modalità di SET-UP. Una volta entrati in questa modalità sullo schermo LCD verranno visualizzati tutti i parametri con le relative impostazioni attuali, facendo uso della tastiera è possibile così effettuare le modifiche desiderate.

Per entrare in modalità SET-UP basta premere ALT-S, da questo momento sia la ricezione che la trasmissione dei caratteri con l'esterno sarà interrotta fino a quando non si uscirà dal SET-UP premendo ESC; non è stato prevista nessuna segnalazione verso il computer remoto per cui durante la fase di SET-UP tutti i caratteri ricevuti saranno persi.

La schermata del terminale durante la fase di SET-UP viene salvata e ripristinata alla sua uscita.

Tutte le impostazioni del terminale vengono salvate automaticamente nel momento in cui si esce da SET-UP, per il loro salvataggio viene utilizzata la memoria EEPROM del microcontrollore, tutte le impostazioni potranno per cui essere recuperate anche nel caso in cui il terminale venga sconnesso dall'alimentazione.

Come anticipato alla prima accensione in assoluto il terminale viene impostato con dei parametri di base, che poi potranno essere modificati dal SET-UP, queste impostazioni di base verranno recuperate anche nel caso in cui si interrompa l'alimentazione durante il salvataggio in EEPROM.

Nella figura 5-1 è riportata la schermata che appare non appena si entra in modalità SET-UP, nella prima colonna sono riportate le varie voci, nella seconda colonna ci sono le relative impostazioni; per spostarsi nei vari campi occorre spostare il puntatore a freccia tramite i tasti cursore UP (↑)e DOWN (↓), una volta selezionata la riga desiderata si può modificare la variabile tramite i tasti cursore LEFT (←) e RIGHT (→), che faranno scorrere le possibili scelte nella colonna delle impostazioni.

Facendo scorrere il puntatore freccia in basso raggiunta l'ultima riga verrà effettuato uno scrolling che farà apparire una nuova riga, e così via finché non sono terminate le righe di SET-UP.

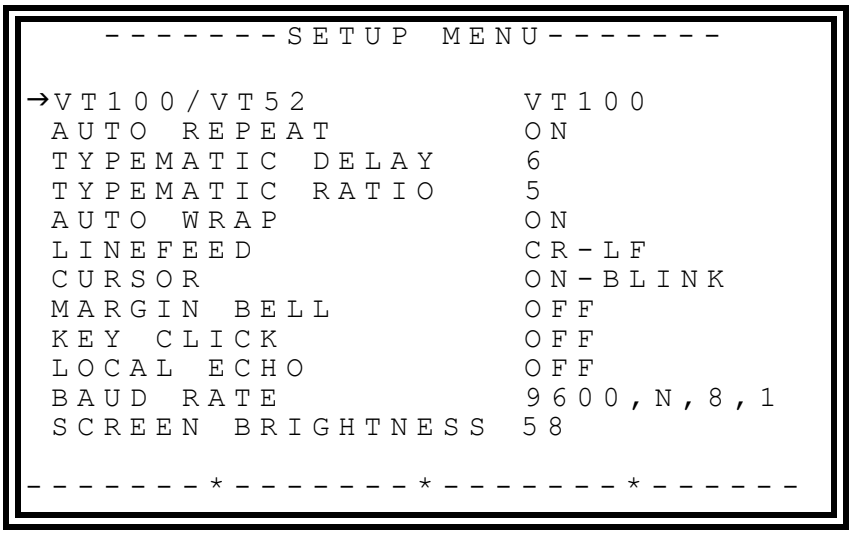


Figura 5-1

Raggiunto l'ultimo campo di SET-UP chiamato TAB STOPS si vedrà nell'ultima riga dello schermo lampeggiare il cursore, è possibile adesso modificare le posizioni dei TAB; in pratica con i tasti cursore LEFT (←) e RIGHT (→) si sposta il cursore lungo tale riga, nello stesso tempo nella colonna delle impostazioni verrà indicata la posizione del cursore, nell'esempio di figura 5-2 il cursore si trova nella colonna 3. Una volta posizionato il cursore nella colonna desiderata premendo la barra spaziatrice (SPACE) si può impostare il TAB STOP a tale posizione, in tal caso al posto del segno '-' apparirà il segno '*', sempre con la barra spaziatrice è possibile togliere il TAB STOP ripremendola alla stessa colonna.

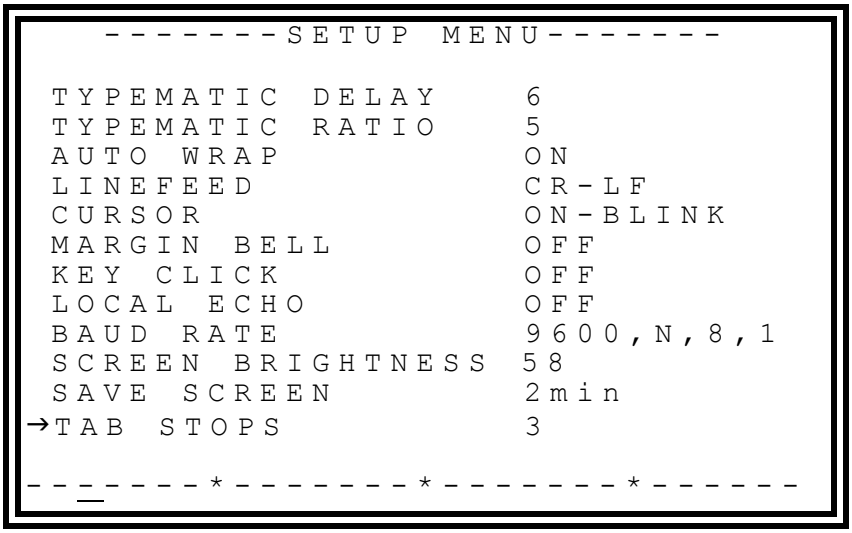


Figura 5-2

Per quanto riguarda il resto dei campi di setup nella tabella sottostante sono riportate tutte le possibili impostazioni di ognuno, nella seconda colonna sono inoltre riportate le impostazioni di base.

Nome del campo	Impostazioni di base	Altre impostazioni	Descrizione delle impostazioni
VT100 / VT52	VT100	VT52	Standard del terminale
AUTO REPEAT	ON	OFF	Auto ripetizione dei tasti ON= abilitato OFF= disabilitato
TYPEMATIC DELAY	6	0:9	Ritardo all'inizio ripetizione
TYPEMATIC RATIO	5	0:9	Velocità di autoripetizione
AUTO WRAP	ON	OFF	A capo automatico a fine riga ON= abilitato OFF= disabilitato
LINEFEED	CR-LF	CR	Azione relativa alla pressione di RETURN e alla ricezione di LF, vedi tabella **Linefeed/Newline mode**
CURSOR	ON-BLINK	ON OFF OFF-BLINK	Attributi del cursore ON= cursore acceso fisso ON-BLINK= cursore acceso lampeggiante OFF= cursore spento OFF-BLINK= Cursore a blocco lampeggiante
MARGIN BELL	OFF	ON	Avviso acustico di margine raggiunto ON= abilitato OFF= disabilitato
KEY CLICK	OFF	ON	Avviso acustico ad ogni pressione di un tasto ON= abilitato OFF= disabilitato
LOCAL ECHO	OFF	ON	Stampa sullo schermo dei

			caratteri trasmessi da tastiera ON= abilitato OFF= disabilitato
BAUD RATE	9600,N,8,1	300:38400,N,8,1 300:38400,E,7,1 300:38400,O,7,1	Selezione del Baud rate RX , TX e attivazione parità pari o dispari
SCREEN BRIGHTNESS	USE POTENZ.	0:99	Regolazione contrasto display USE POTENZ.= Utilizzare il potenziometro esterno 0:99= Utilizzare tasti cursore
SAVE SCREEN	OFF	2 min 5 min 10 min 30 min 60 min	Salva schermo, spengimento della retro illuminazione OFF= Disattivato 2:60 min = Tempo di intervento variabile
TAB STOPS	1	1:30	Colonna dove inserire o togliere un TAB STOP

5.2 Pulizia dello schermo:

In molti casi durante l'uso del terminale può risultare utile eseguire una pulizia totale dello schermo, per cui è stata prevista una combinazione di tasti ALT-S che oltre a cancellare tutto il display posiziona il cursore all'angolo superiore sinistro. Questa funzione non riguarda lo standard VT100, ma è stata aggiunta solo per comodità, infatti secondo lo standard VT100 la pulizia dello schermo deve avvenire solo tramite la ricezione di una sequenza trasmessa dal computer remoto, (vedi **6.2 Caratteri e sequenze riconosciuti dal terminale**).

6. APPENDICE

6.1 Caratteri trasmessi dal terminale

Sono qui di seguito riportate le varie tabelle dei caratteri o delle sequenze trasmesse dai tasti della tastiera.

Nella seguente tabella sono riportati tutti i caratteri di controllo generati dal terminale, tutti questi codici (escluso DELete) possono essere generati tramite la pressione contemporanea di alcuni tasti con CTRL, alcuni di essi invece possono essere generati direttamente da tasti dedicati.

Caratteri di controllo generati

Nome	**Mnemonico**	**Codice spedito (decimale)**	**Tasto premuto con CTRL**	**Tasto dedicato**
Null	NUL	0	**2**	
Start of heading	SOH	1	**A**	
Start of text	STX	2	**B**	
End of text	ETX	3	**C**	
End of transmission	EOT	4	**D**	
Enquire	ENQ	5	**E**	
Acknowledge	ACK	6	**F**	
Bell	BEL	7	**G**	
Backspace	BS	8	**H**	**BACK SPACE**
Horizontal tab	HT	9	**I**	**TAB**
Linefeed	LF	10	**J**	
Vertical tab	VT	11	**K**	
Form feed	FF	12	**L**	
Carriage return	CR	13	**M**	**ENTER***
Carriage return linefeed	CR LF	13 10		**ENTER ***
Shift out	SO	14	**N**	
Shif in	SI	15	**O**	
Data link escape	DLE	16	**P**	
Device control 1	DC1 (XON)	17	**Q**	

Device control 2	DC2	18	**R**	
Device control 3	DC3 (XOFF)	19	**S**	
Device control 4	DC4	20	**T**	
Negative acknowledge	NAK	21	**U**	
Synchronous idle	SYN	22	**V**	
End of transmission block	ETB	23	**W**	
Cancel previous word or character	CAN	24	**X**	
End of medium	EM	25	**Y**	
Substitute	SUB	26	**Z**	
Escape	ESC	27	**[**	
File separator	FS	28	\	
Group separator	GS	29	**]**	
Record separator	RS	30	**6**	
Unit separator	US	31	**-**	
Delete	DEL	127		**DELETE**

*Il carattere corrispondente al tasto **ENTER** può essere cambiato dall'impostazione Linefeed / new line mode. Quando è off **ENTER** genera un singolo carattere di controllo (CR), quando è on genera due caratteri (CR,LF).

Impostazione Linefeed / New line mode

Selezione	**Tasto premuto-carattere spedito**	**Carattere ricevuto-azione eseguita**
OFF	RETURN - CR	CR - Sposta cursore margine sinistro
OFF	CTRL+J - LF	LF - Sposta il cursore una riga sotto, mantendo la stessa colonna
ON	RETURN - CR LF	CR - Sposta cursore margine sinistro
ON	CTRL+j - LF	LF - Sposta il cursore al margine sinistro della riga sottostante

Nella seguente tabella sono riportati i caratteri trasmessi dalla tastierina numerica, con i tasti impostati numerici, ossia NUM-LOCK attivo; come si osserva i caratteri trasmessi dipendono

dalla modalità Application o Normal che può essere impostata solo tramite un apposita sequenza ricevuta dall'esterno (vedi caratteri ricevuti).

Caratteri e sequenze trasmesse dalla tastierina numerica

	VT100		VT52	
Tasto	**Numeric Keypad Mode**	**Application Keypad Mode**	**Numeric Keypad Mode**	**Application Keypad Mode**
0	0	ESC O p	0	ESC ? p
1	1	ESC O q	1	ESC ? q
2	2	ESC O r	2	ESC ? r
3	3	ESC O s	3	ESC ? s
4	4	ESC O t	4	ESC ? t
5	5	ESC O u	5	ESC ? u
6	6	ESC O v	6	ESC ? v
7	7	ESC O w	7	ESC ? w
8	8	ESC O x	8	ESC ? x
9	9	ESC O y	9	ESC ? y
+	+	ESC O l	+	ESC ? l
-	-	ESC O m	-	ESC ? m
.	.	ESC O n	.	ESC ? n
*	*	ESC O M	*	ESC ? M
ENTER*	CR o CR LF	CR o CR LF	CR o CR LF	CR o CR LF

***Il tasto ENTER della tastierina numerica dipende dalla modalità LINEFEED / NEW LINE, nel caso sia attiva verranno generati due caratteri CR-LF, altrimenti il singolo carattere CR.

Quando il BLOC-NUM è disattivo i tasti della tastierina numerica cambiano funzione, alcuni di essi diventano tasti cursore; come si osserva dalla tabella solo per lo standard VT100 si può avere la modalità SET o RESET questa verrà impostata dall'esterno tramite un apposita sequenza, (vedi caratteri ricevuti). I tasti HOME e END a differenza degl'altri non dipendono dalla modalità Cursor Key Mode.

Sequenze generate dai tasti cursore

	VT100		VT52
Cursor Key	**Cursor Key Mode reset**	**Cursor Key Mode set**	**Cursor Key**
↖ (HOME)	ESC [H	ESC [H	ESC H
↑	ESC [A	ESC O A	ESC A
↓	ESC [B	ESC O B	ESC B
→	ESC [C	ESC O C	ESC C
←	ESC [D	ESC O D	ESC D
END	ESC [K	ESC [K	ESC K

La tastiera standard VT100-52 prevede sopra ai tasti della tastierina numerica quattro tasti funzione PF1:PF4, dato che in questo terminale si è utilizzato una tastiera PC si è dovuto simulare la loro presenza tramite le combinazioni SHIFT F1:F4, nella seguente tabella sono riportati i caratteri trasmessi da ciascuna combinazione.

Caratteri trasmessi dai tasti funzione

Tasto LCDTERM	**Tasto equivalente VT100**	**VT100 Mode**	**VT52 Mode**
Shift-F1	PF1 key	ESC O P	ESC P
Shift-F2	PF2 key	ESC O Q	ESC Q
Shift-F3	PF3 key	ESC O R	ESC R
Shift-F4	PF4 key	ESC O S	ESC S

6.2 Caratteri e sequenze riconosciuti dal terminale

Gli standard VT100 VT52 riconoscono alcuni dei caratteri di controllo, come anticipato questi non dovranno essere stampati sullo schermo, ma eseguire determinate azioni. La seguente tabella

descrive le funzioni relative ad ogni carattere di controllo riconosciuto, alcuni di questi non sono stati implementati, dato che il terminale LCD non possiede tutte le caratteristiche, quali la scelta fra più set di caratteri.

Caratteri di controllo ricevuti

Nome	**Mnemonico**	**Codice ricevuto (decimale)**	**Funzione da eseguire**
Null	NUL	0	Ignorato quando ricevuto
Enquire	ENQ	5	Non implementato
Bell	BEL	7	Genera avviso acustico
Back space	BS	8	Sposta il cursore a sinistra di un carattere, finchè non raggiunge il margine sinistro, in questo caso non esegue nessuna azione
Horizontal tab	HT	9	Sposta il cursore al prossimo tab stop, o al margine destro se non ci sono più tab stops
Linefeed	LF	10	Vedi tabella **Linefeed/Newline mode**
Vertical tab	VT	11	Processato come LF
Form feed	FF	12	Processato come LF
Carriage return	CR	13	Sposta il cursore al margine sinistro della linea corrente
Shift out	SO	14	Non implementato
Shift in	SI	15	Non implementato
Device Control 1	DC1	17	Non implementato
Device control 3	DC3	18	Non implementato
Cancel	CAN	24	Cancella la sequenza ricevuta sia di controllo che di escape
Substitute	SUB	26	Processato come CAN
Escape	ESC	27	Processato come introduttore di sequenza

Sequenze di escape e di controllo

Le seguenti sequenze di controllo hanno la funzione di settare delle impostazioni del terminale, alcune di esse possono essere modificate anche in fase di setup, (vedi **5.1 Modalità SET-UP**). Si

distinguono due tipi di sequenze, Set Mode e Reset Mode che differiscono dal carattere terminale ‘h’ o ‘l’, per questo tipo di sequenze è possibile attivare o disattivare anche più di una modalità alla volta, inserendo il carattere separatore ‘;’ fra un parametro e l’altro della sequenza, ad esempio:

ESC [? 6 ; 7 h , questa sequenza imposta attivi sia l’autowrap che l’autorepeat.

La modalità origin mode, permette di attivare una regione di scrolling ristretta rispetto all’intero schermo, tale regione di scrolling viene definita da un apposita sequenza descritta più avanti (vedi tabella **regione di scrolling**). Quando questa modalità è attiva la Home Position diventa l’estremo superiore della regione di scrolling, la posizione del cursore diventa per cui relativa a tale punto; quando invece la modalità origin mode diventa absolute, allora la posizione del cursore è riferita all’estremo superiore dello schermo LCD.

In pratica l’attivazione o meno di questa modalità fa divenire i limiti superiore e inferiore dello schermo ai quali avviene lo scrolling, o i margini estremi del display oppure i margini della regione di scrolling.

Sequenze di settaggio impostazioni terminale Set Mode

Nome	**Mnemonico**	**modalità**	**sequenza**
Linefeed / new line	LMN	New line	ESC [2 0 h
Cursor key	DECCKM	Set	ESC [? 1 h
VT100 / VT52	DECANM	VT100	
Origin	DECOM	Relative	ESC [? 6 h
Autowrap	DECAWM	On	ESC [? 7 h
Auto repeat	DECARM	On	ESC [? 8 h

Sequenze di settaggio impostazioni terminale reset Mode

Nome	**Mnemonico**	**modo**	**sequenza**
Linefeed / new line	LMN	Linefeed	ESC [2 0 l*
Cursor key	DECCKM	Reset	ESC [? 1 l*
VT100 / VT52	DECANM	VT52	ESC [? 2 l*
Origin	DECOM	Absolute	ESC [? 6 l*
Autowrap	DECAWM	Off	ESC [? 7 l*
Auto repeat	DECARM	Off	ESC [? 8 l*

* L’ultimo carattere della sequenza è una L minuscola.

Come anticipato la tastierina numerica quando il NUM-LOCK è attivo può generare sia caratteri numerici che sequenze (vedi caratteri trasmessi), per commutare lo stato si utilizzano le seguenti sequenze. La selezione fra modalità application e numeric non effettua nessuna modifica ai tasti cursore nel caso in cui il NUM-LOCK sia disattivo.

Sequenze di impostazione modalità tastierina numerica

Nome	**Mnemonico**	**Sequenza**
Application	DECKPAM	ESC =
Numeric	DECKPNM	ESC >

Lo standard VT100 prevede la definizione di una porzione di schermo di grandezza variabile, nella quale è possibile eseguire lo scrolling sia up che down, tramite la seguente sequenza viene impostato il margine superiore (top) alla riga Pt ed il margine inferiore (bottom) alla riga Pb.
Nel caso in cui la regione di scrolling non venga definita allora diviene automaticamente tutta la schermata fisica del display.

Regione di scrolling

Nome	**Mnemonico**	**Sequenza**	**descrizione**
Set top and bottom margin	DECSTBM	ESC [Pt ; Pb r	Imposta margini regione di scrolling Pt = valore riga top Pb = valore riga bottom

Le seguenti sequenze si occupano tutte del posizionamento del cursore nello schermo; il parametro P? è un numero decimale espresso in forma ASCII, nel caso in cui non venga specificato oppure valga 0 il terminale assumerà il parametro uguale a 1.
Quando la modalità origin mode è impostata relative, allora il cursore non può uscire dalla regione di scrolling, a meno che non vengano utilizzate le sequenze CUP e HVP; tramite le quali si indicano direttamente le coordinate (riga, colonna) dove posizionare il cursore, tali coordinate sono riferite alla Home position.
La Home position nel caso in cui la modalità origin mode sia relative sarà l’estremo sinistro superiore della regione di scrolling.

Posizionamento cursore

Nome	Mnemonico	sequenza	descrizione
Cursor up	CUU	ESC [Pn A	Sposta il cursore in alto di Pn righe
Cursor down	CUD	ESC [Pn B	Sposta il cursore in basso di Pn righe
Cursor forward (right)	CUF	ESC [Pn C	Sposta il cursore a destra di Pn colonne
Cursor backward (left)	CUB	ESC [Pn D	Sposta il cursore a sinistra di Pn colonne
Cursor position	CUP	ESC [Pr ; Pc H	Posiziona il cursore alla riga Pr e alla colonna Pc, riferiti a Home position.
		ESC H	Posiziona il cursore a Home Position
Horizontal and vertical position	HVP	ESC [Pr ; Pc f	Posiziona il cursore alla riga Pr e alla colonna Pc
		ESC [f	Posiziona il cursore a Home Position
Index	IND	ESC D	Porta il cursore alla riga sottostante, stessa colonna; raggiunto il margine inferiore effettua lo scroll up
Reverse index	RI	ESC M	Porta il cursore alla riga sovrastante, stessa colonna; raggiunto il margine superiore effettua lo scroll down
Next line	NEL	ESC E	Sposta il cursore alla prima colonna della riga sottostante
Save cursor (and attributes)	DECSC	ESC 7	Salva la posizione del cursore e l'impostazione

			origin mode
Restore cursor (and attributes)	DECRC	ESC 8	Ripristina la precedente posizione del cursore salvata, e l'impostazione origin mode

La posizione dei tab stops può essere modificata oltre che da Set-up anche dalle seguenti sequenze.

Tab stops

Nome	**Mnemonico**	**sequence**	**descrizione**
Horizontal tab set	HTS	ESC H	Aggiunge un tab stop alla posizione del cursore
Tabulation clear	TBC	ESC [g ESC [0 g	Cancella il tab stop alla posizione del cursore
		ESC [3 g	Cancella tutti i tab stops

Le seguenti sequenze rimuovono i caratteri dallo schermo, una volta cancellati i caratteri vengono persi. La posizione del cursore non viene modificata.

Cancellazione caratteri

Nome	**Mnemonico**	**sequenza**	**descrizione**
Erase in line	EL	ESC [K ESC [0 K	Cancellano i caratteri dello schermo a partire dal cursore fino alla fine della riga
		ESC [1 K	Cancella dall'inizio riga alla posizione cursore
		ESC [2 K	Cancella tutta la riga
Erase in display	ED	ESC [J	Cancellano dal

		ESC [0 J	cursore fino alla fine dello schermo
		ESC [1 J	Cancella dall'inizio schermo fino alla posizione cursore
		ESC [2 J	Cancella tutto lo schermo

Le seguenti sequenze permettono al computer remoto di conoscere lo stato del terminale, il tipo di terminale e la posizione del cursore. Si avranno per cui sequenze di richiesta (request) ricevute dal terminale, e sequenze di risposta (response) trasmesse dal terminale.

Reports

Nome	**Mnemonico**	**sequenza**	**descrizione**
Device status report (request)	DSR	ESC [5 n	Richiesta da parte del computer di riferire lo stato del terminale
(response)		ESC [0 n	Risposta di terminale Ok
(response)		ESC [3 n	Risposta di terminale malfunzionante
(request)		ESC [6 n	Richiesta di riferire la posizione del cursore
Cursor position report (response)	CPR	ESC [Pr ; Pc R	Risposta alla richiesta di

			posizione cursore Pr = Riga Pc = Colonna
Device attributes (request)	DA	ESC [c ESC [0 c	Richiesta di identificazione del terminale
Identify the terminal (request)	DECID	ESC Z	Richiesta di identificazione del terminale tramite sequenza DA
Device attributes (response)	DA	ESC [? 1 ; 0 c	Risposta di terminale VT100

La seguente sequenza porta il terminale nello stato iniziale, vengono ripristinate per cui tutte le impostazioni che si hanno all'accensione. La tastierina numerica viene impostata in numeric mode, i tasti cursore in reset mode, l'origin mode viene impostata absolute. I margini della regione di scrolling diventano l'intera area del display.

Reset

nome	**Mnemonico**	**sequenza**	**descrizione**
Reset to initial state	RIS	ESC c	Genera un reset riportando il terminale allo stato iniziale (d'accensione)

Standard VT52

Il passaggio alla modalità VT52 può essere effettuato oltre che da Set-Up, anche dall'esterno tramite apposita sequenza (vedi standard VT100). Quando viene effettuato il passaggio dallo standard VT100 al VT52, molte impostazioni che erano state precedentemente settate non possono più essere modificate dall'esterno, dato che il VT52 possiede un set di sequenze più ristretto; per modificare alcuni parametri si può comunque agire tramite Set-up, ad esempio per i tab stops.

Le uniche modalità che possono essere modificate sono le seguenti.

Set and Reset Mode

nome	Mnemonico	sequenza	descrizione
Enter VT100 mode	ENVT100	ESC <	Passa a standard VT100
Enter application keypad mode	DECKPAM	ESC =	Imposta tastierina numerica in application mode
Exit application keypad mode	DECKPNM	ESC >	Imposta tastierina numerica in modalità normale

Le seguenti sequenze permettono di spostare il cursore, i margini utilizzati rimangono gli stessi della modalità VT100, infatti in VT52 non è possibile modificare i margini della regione di scrolling.

La sequenza DRA è l'unica a possedere dei parametri, Nr ed Nc, ciascun parametro è espresso sotto forma di un unico carattere, dal quale occorre sottrarre 32 decimale per ricavare il vero valore.

Cursor position

nome	Mnemonico	sequenza	descrizione
Cursor up	CUU	ESC A	Cursore su di una riga
Cursor down	CUD	ESC B	Cursore giù di una riga
Cursor right	CUF	ESC C	Cursore a destra di una colonna
Cursor left	CUB	ESC D	Cursore a sinistra di una colonna
Cursor to home	CUP	ESC H	Cursore a home position
Direct cursor address	DRA	ESC Y Nr Nc	Posizionamento cursore a: Riga = Nr-32 Colonna = Nc-32

Reverse line feed	RI	ESC I	Sposta il cursore su di una riga se raggiunge il margine superiore viene effettuato lo scroll down

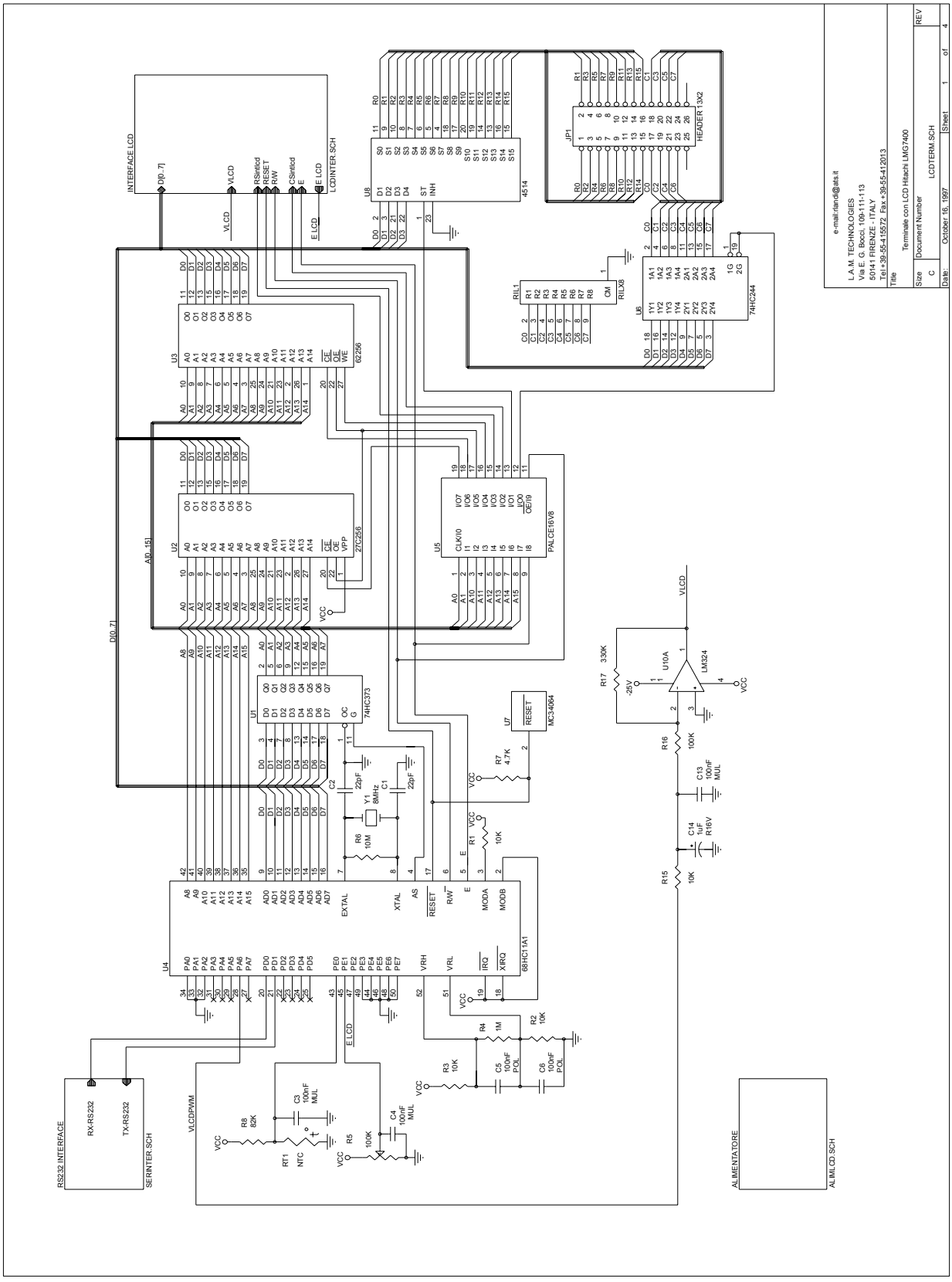
Erasing

Nome	**Mnemonico**	**sequenza**	**descrizione**
Erase to end of line	EL	ESC K	Cancella tutti i caratteri, a partire dal cursore compreso fino alla fine della riga.
Erase to end of screen	ED	ESC J	Cancella tutti i caratteri a partire dal cursore compreso fino alla fine dello schermo

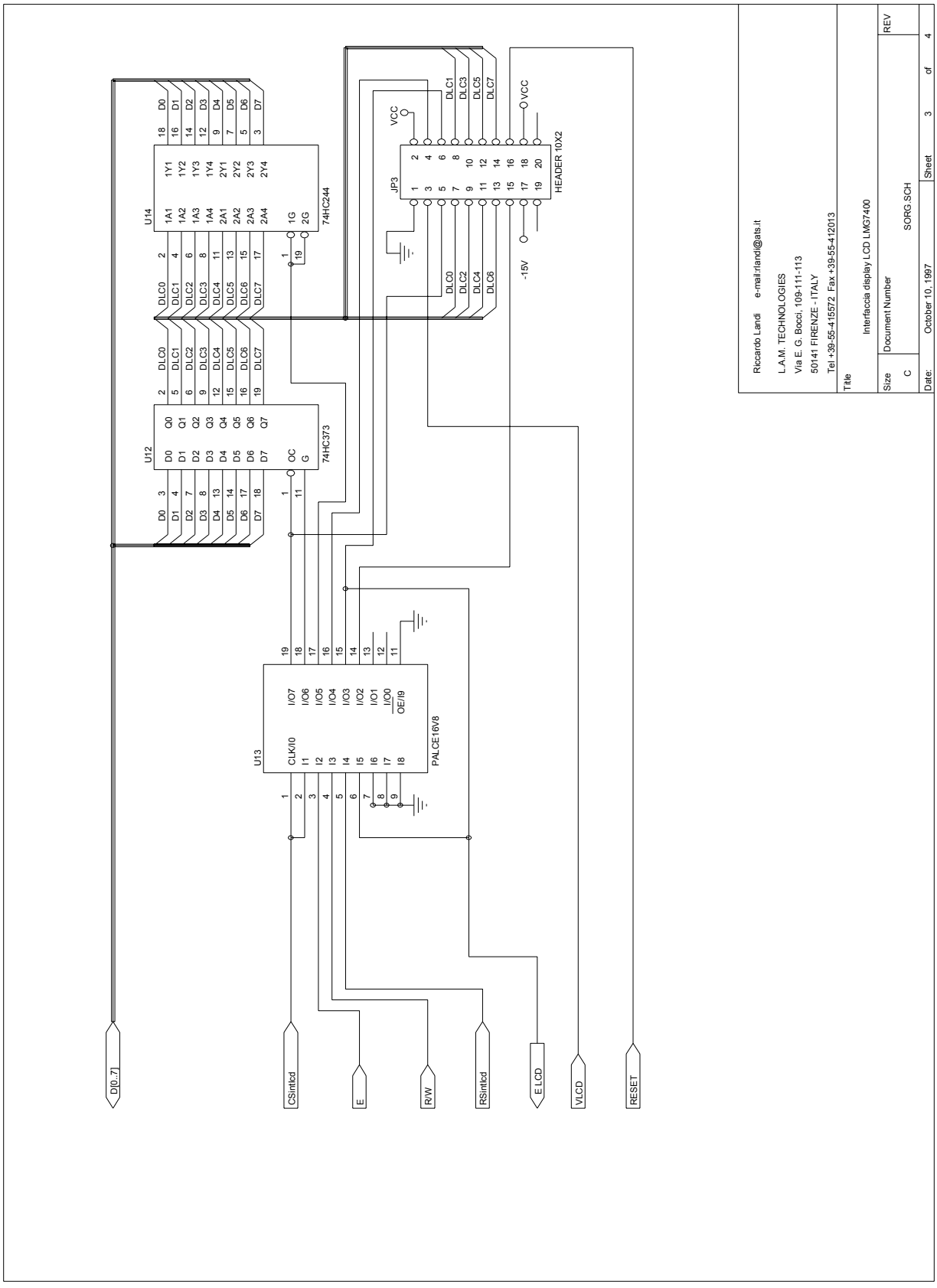
Reports

Nome	**Mnemonico**	**sequenza**	**descrizione**
Identify (what are you)	DECID	ESC Z	Richiesta di identificazione del terminale
Response: VT100 e VT52	IDVT52	ESC / Z	Risposta alla richiesta di identificazione, valida sia per VT100 che

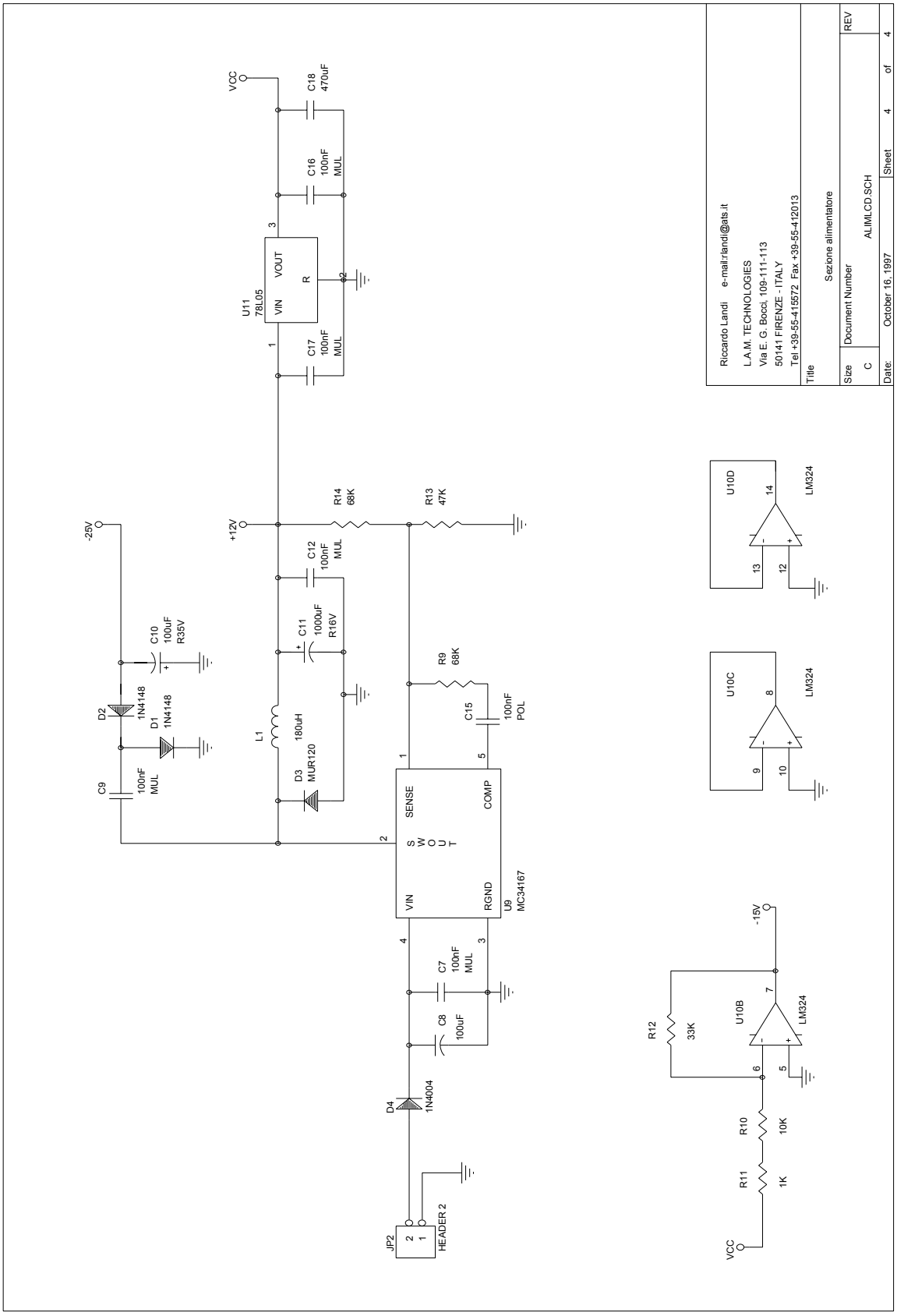
6.3 Schema elettrico terminale LCD



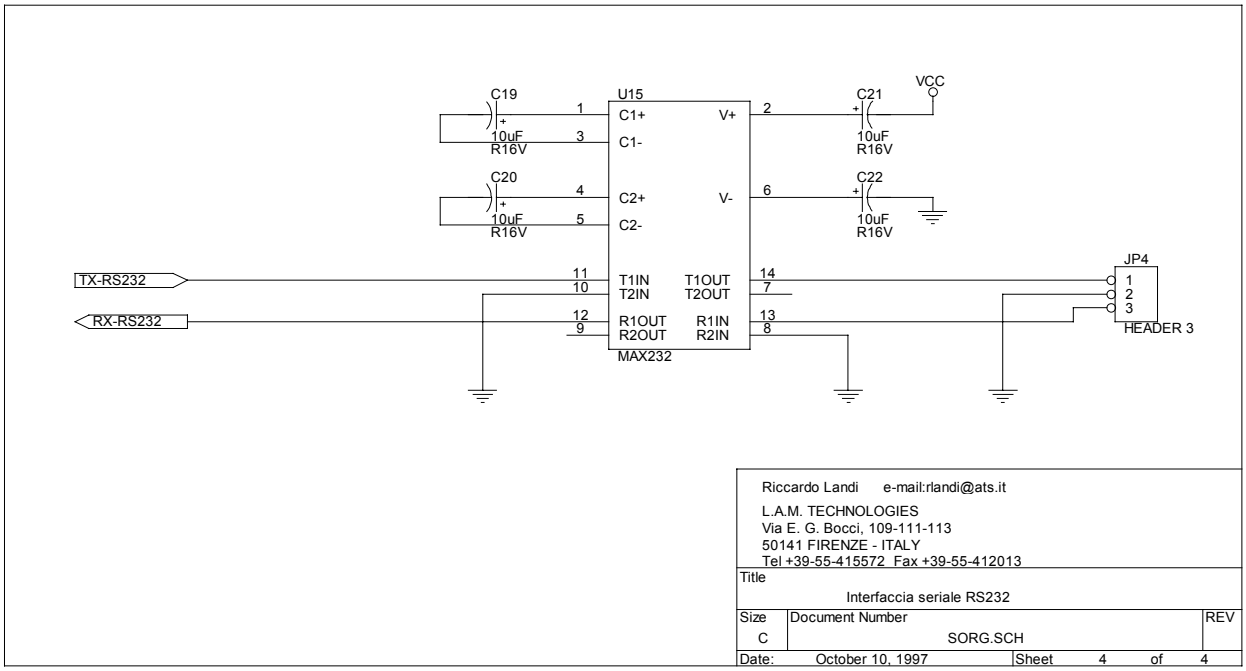
6.4 Schema elettrico interfaccia display LCD



6.5 Schema elettrico alimentatore



6.6 Schema elettrico interfaccia seriale RS232



7. BIBLIOGRAFIA

- LIQUID CRYSTAL GRAPHIC Display modules, HITACHI
- HC11, M68HC11 A series, MOTOROLA
- HITACHI IC Memory No.1, Marzo 1992
- PAL Device Data Book and Design Guide, AMD
- Analog / Interface ICs Device Data Vol I, MOTOROLA
- ICC IARC compiler, IAR SYSTEM
- VT100 EMULATION http://www.nw.com/nw/WWW/products/wizcon/vt100.html
- VT52 EMULATION http://www.nw.com/nw/WWW/products/wizcon/vt52.html
- DIGITAL VT101 TERMINAL OPERATION MANUAL
- AMPEX 219 VIDEO DISPLAY TERMINAL OPERATION MANUAL